A NEW PARAMETER OPTIMIZATION METHOD APPLIED TO AUTOPILOT DESIGN

M. A. S. MacNamara

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A NEW PARAMETER OPTIMIZATION METHOD
APPLIED TO AUTOPILOT DESIGN

by

M. A. S. MacNamara

March 1975

Thesis Advisor: George J. Thaler

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A New Parameter Optimization Method Applied to Autopilot Design	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; March 1975
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): M. A. S. MacNamara	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: March 1975
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Naval Postgraduate School, Monterey, California 93940	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

** Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An investigation of a method for finding an optimum compensator for use in an autopilot is described. Given a structure with a number of real poles and zeros which is steady-state decoupled, the pole-zero locations are chosen to give a system response as close as possible to a desired response. The desired response can be any one or a combination of system outputs. The computer is used to optimize the pole-zero locations by means of a func-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

tion minimization program.

A New Parameter Optimization Method Applied to Autopilot Design

by

M. A. S. MacNamara
Major, Canadian Forces
Royal Military College, 1961
BASc, University of Toronto, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

ABSTRACT

An investigation of a method for finding an optimum compensator for use in an autopilot is described. Given a structure with a number of real poles and zeros which is steady-state decoupled, the pole-zero locations are chosen to give a system response as close as possible to a desired response. The desired response can be any one or a combination of system outputs. The computer is used to optimize the pole-zero locations by means of a function minimization program.

TABLE OF CONTENTS

I. INTRODUCTION

The autopilot is used for aircraft control to hold some variable of the aircraft's motion constant, such as speed, altitude or glide slope angle, in the presence of disturbances. These disturbances can originate outside the aircraft, e.g. random wind gusts, or within the system such as electrical noise in actuators and control system components.

The problem becomes more complex when the aircraft is viewed as a multivariable system, i.e. more than one input and output. For example, a movement of the elevator control will change the pitch angle and the speed, and a change in thrust likewise. It would be desirable for precise control to have one input cause a change in a corresponding output and in no other. This condition would require that the transfer function matrix of the entire system be a diagonal matrix, which allows much less freedom in designing for stability.

This condition can be improved by requiring that the non-change in other variables (decoupling) occur only after the initial transients have died out, that is, once steady state has been reached.

Methods have been devised to obtain steady state decoupling by linear state-variable feedback and by cascade compensation with unity feedback (Huang).

In particular, for the cascade compensation case, which is probably more practical than state variable feedback, Huang

found that the rank condition for the plant matrix is no longer necessary as it was in the state variable feedback case and that pure integrators in the compensators and poles at the origin in the plant transfer function were helpful in providing steady-state decoupling.

However, when it comes to designing the remainder of the compensators for stability or a desired transient response, one is left with a tedious root-locus design process which to some extent requires considerable experience in this technique. The problem is aggravated by the fact that the root loci for each channel are interdependent and this complexity increases with the number of inputs and outputs. Even when completed, there is no guarantee that the design is an optimum one.

The intent of this thesis is to develop a method of applying the power of the modern digital computer and the techniques of linear programming to the process of compensator design. Essentially the process consists of simulating the system in the time domain with the poles and zeros of the compensators as unknowns. By having a function minimization program minimize the integral-squared-difference between the response of the simulated system and a desired response, the required poles and zeros are determined.

Cantalapiedra used a similar method with the same function minimization program to determine the optimum number of

poles and zeros to be used in a model of a high order system. In his thesis the integral squared difference between the system and model response curves was minimized for various configurations of the model and this so-called cost function served as a measure of how well the low-order model approximated the high-order system.

Lima, in his thesis, used the same function minimization program to obtain the gains for a compensated ship controller in a model of two ships in underway replenishment. In this case a desired track was to be followed by one ship with reference to the replenishing ship in the presence of surge and sway forces affecting both ships.

II. DESCRIPTION OF THE SYSTEM

In this thesis, the longitudinal motion of the C-8A Augmentor Wing "Buffalo" aircraft presently under test and development at the NASA Ames Research Center was modelled. There are two reasons for this choice. First, a STOL aircraft such as the augmentor wing Buffalo, in the landing or take-off phase, operates at low speeds and steep flight path angles and in such flight regimes a strong artificial stability must be applied just to make the aircraft flyable. Second, a model of this same aircraft was used by Huang in his investigation of decoupling referenced earlier; thus the plant equations were readily available as well as the compensator transfer functions which he determined by the root-locus method.

Only the longitudinal motion of the aircraft was investigated since the principles arrived at for this mode would be equally applicable to the lateral-directional mode in a more complicated model. The aircraft is initially considered as flying straight and level at a speed V_o = 126.7 ft/sec when it is subjected to a reference pitch angle step input of -0.25 radians at time = 0.0. These conditions apply to all aspects of the investigation which follows. Ideally, if the system were perfectly decoupled, the pitch angle would change to -0.25 radians and the speed and angle of attack would remain at their trim conditions. However,

as will be noted, there is a short period of transients which follows the initial step input which die out leaving speed and angle of attack at their trim conditions and the pitch angle at -0.25 radians.

Figure II-1 shows a schematic diagram of the system model consisting of the plant dynamics and the actuator dynamics and cascade compensators; one for each channel. The three inputs to the plant are δ_t , % thrust (trim = 100%); δ_f , flap angle deflection from trim in radians and δ_e , elevator deflection from trim in radians. The three outputs from the plant are u , the speed variation from trim in ft/sec, α , the angle of attack in radians, and θ, the pitch angle in radians. The feedback of outputs to inputs is to a certain extent arbitrary, however by inspection one could feedback an output to the input which had the greatest effect on it. In any case, the scheme adopted here is that used by Huang in his thesis.

Figure II-2 shows a schematic diagram of the system model as designed by Huang. The compensator poles and zeros which he determined by root-locus techniques, the actuator transfer functions and the plant equations are also shown. The compensator poles at the origin, necessary for decoupling, should be noted. The response of this model to a pitch angle step input of -0.25 radians is shown in Figures II-3, II-4 and II-5 for speed, angle of attack and pitch angle respectively. These responses are used as reference responses during the course of the investigation. It will be noted

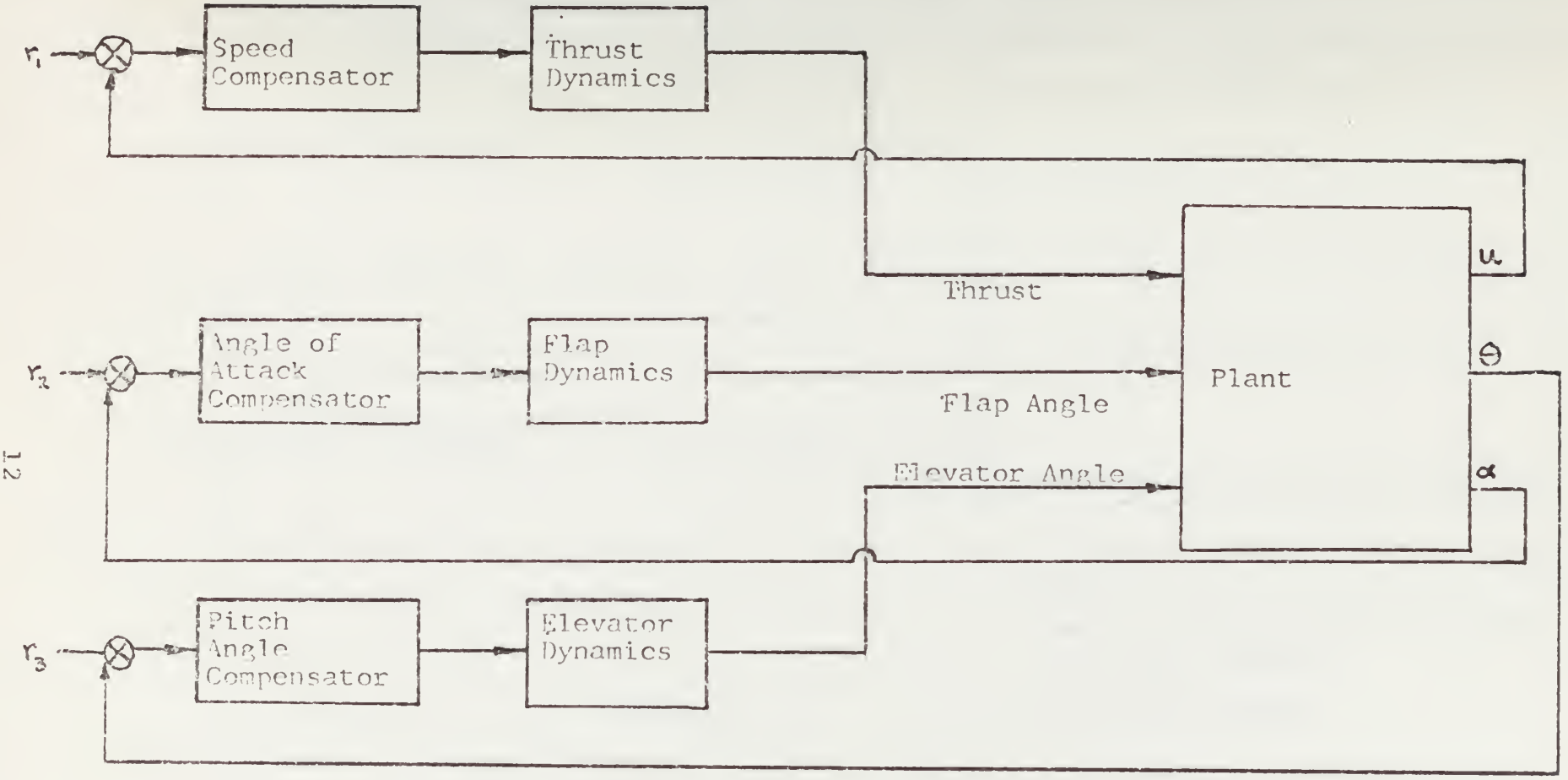
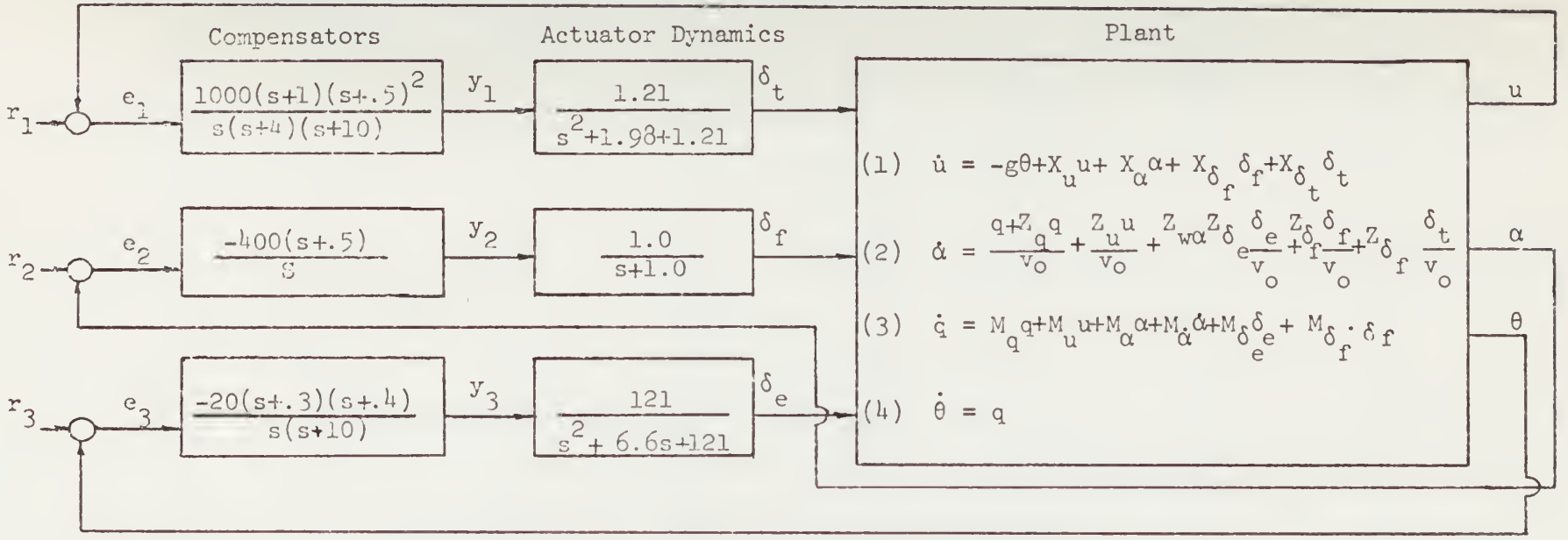

r_1
Speed Compensator
Thrust Dynamics
Thrust
u_1
r_2
Angle of Attack Compensator
Flap Dynamics
Flap Angle
Plant
Θ
Elevator Angle
α
r_3
Pitch Angle Compensator
Elevator Dynamics

Figure II - 1



u = speed difference from V_o (126.7 ft/sec)

α = angle of attack (radians) from trim

θ = pitch angle (radians)

q = pitch rate (rad/sec)

FIGURE II - 2

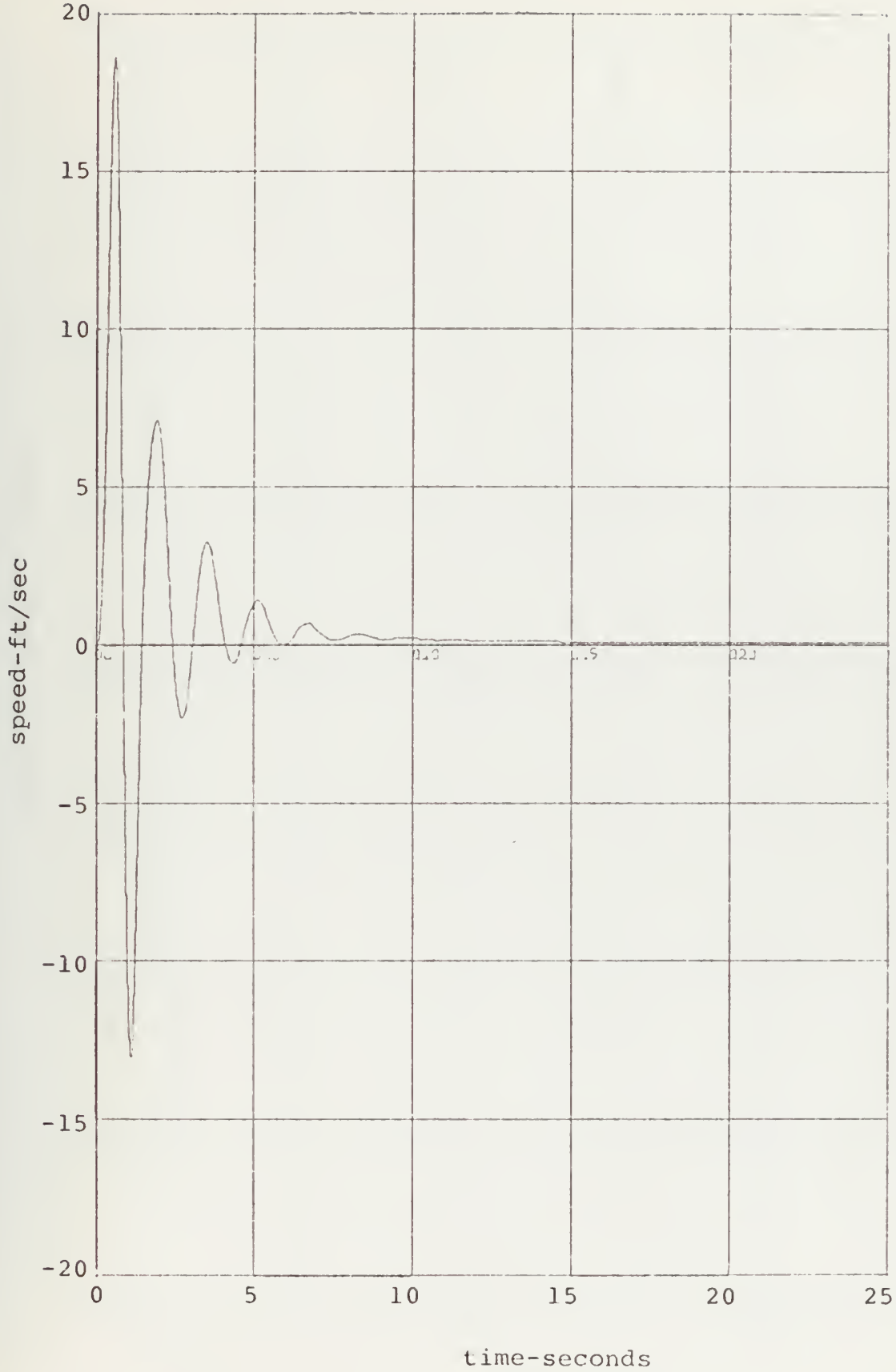


Figure II - 3

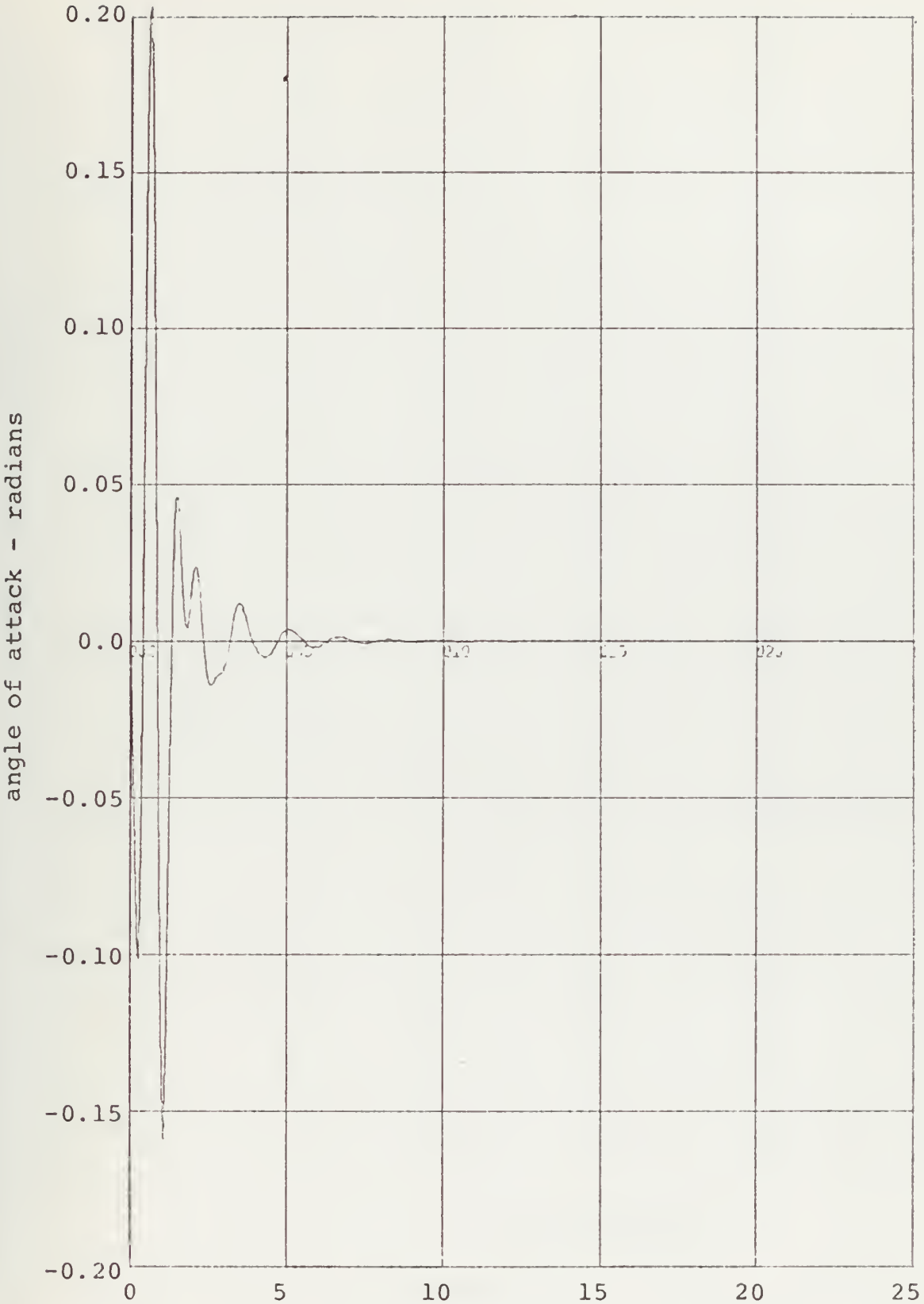

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure II - 4

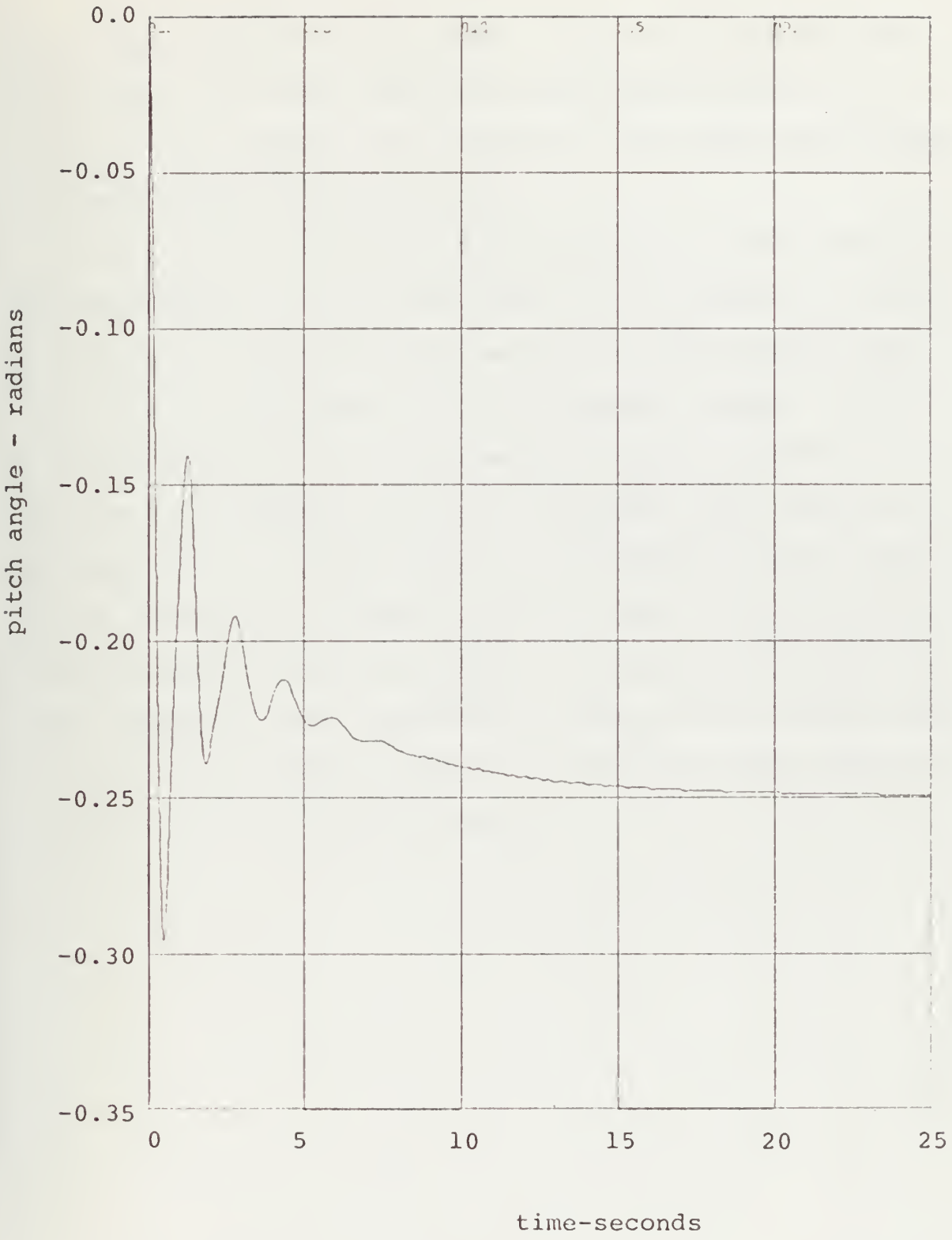

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure II - 5

that these responses are somewhat oscillatory and of large amplitude. One major intent of this investigation is to improve (i.e., reduce) the frequency, magnitude and duration of these transient oscillations.

It is assumed for this investigation that the structure of the compensators is fixed; that is, the numbers of poles and zeros in each compensator will be as designed by Huang. This is not to say that this is an optimum number. Indeed a greater number of poles and zeros in each compensator may improve the response over what is possible with the present configuration. A lesser degree of complexity and an interest in determining how much could be done with the present configuration dictated the choice. Similarly, the gains in each compensator are considered as fixed at the values shown in Figure II-2, though there is no reason why they could not be made a further three variables.

III. TECHNIQUE FOR SOLUTION

A. THE COST FUNCTION

The point in question for the investigation is to determine whether and at what cost a given 3X3 plant can be made to follow a desired output response by determining the poles and zeros of cascade compensators of a given configuration using the digital computer. When given a curve of the desired output response, the most obvious measure of how closely the model has matched it, is the integral-squared-difference between the desired output and the actual output between $t = 0.0$ and some final time t_f. This so-called cost function J has the form:

$$J = \int_{0}^{t_f} \sum_{i=1}^{n} w_i (y_{di} - y_{ai})^2 dt$$

where $y_{di} \equiv$ the desired i th output at time t

$y_{ai} \equiv$ the actual i th output at time t

$w_i \equiv$ weighting factor for i th output

The w_i can be chosen to penalize a more important output deviation over a less important one or, as in the case of the present investigation, to give equal weight to all outputs which vary due to differences in units. It is not necessary that the cost function be a function of all outputs. It could be a function of the one of immediate importance with subsequent checking of the other outputs to follow.

Both types of cost functions were used in this investigation.

The choice of t_f is an important one. If t_f is too long, the cost of the calculations in computer time becomes prohibitive, especially if the time steps used in integration are small. On the other hand if t_f is too short, the results may indicate a good following of the desired response for $t < t_f$, but the system may diverge and go unstable for $t > t_f$. A compromise was reached for this investigation choosing two times; a t_f = 25 seconds, the approximate settling time of the original theta response (see Figure II-5), and t_f = 12.5 seconds. The calculations for curve matching were done over the shorter period and the results checked by simulating the system and plotting the responses over the longer period.

Thus it will be seen that J is a direct measure of how well the sytem output response curve matches a desired output response curve over the simulation time chosen. If J reaches a minimum it can be assumed that, barring the chance of a local rather than a global minimum, the "best" combination of poles and zeros has been found. Alternately, even though a minimum has not been found, a visual comparison of the two responses may indicate that the poles and zeros determined are close enough to an optimum set.

B. THE FUNCTION MINIMIZATION SUBROUTINE

1. General

The function minimization subroutine chosen for this

investigation is one which has seen some use at the Naval Postgraduate School in a variety of disciplines. At the time the present work was proceeding it was undergoing the incorporation of some improvements and the author had the benefit of this more up-to-date version.

The subroutine is called BOXPLX and was programmed by R. R. Hilleary of the Naval Postgraudate School Computer Center. It will find the minimum of any arbitrary function, linear or non-linear subject to explicit constraints on the variables or implicit constraints on functions of the variables. It will handle a maximum of 25 variables, independent and auxiliary. With the subroutine must be appended two user-generated function subroutines to calculate the cost function and check for implicit constraint violations. A brief description of BOXPLX is included at Appendix A to allow a better understanding of some of the processes which follow and to guide further workers in this area.

2. Constraints and Start Points

The variables in BOXPLX are allowed to move within a feasible region (n-dimensional space, where n = number of variables) defined by upper and lower bounds on their values. These values are given as input to BOXPLX for any given problem.

It was assumed that all poles and zeros would be in the left half plane for reasons of stability and thus an absolute lower bound of 0.0 was set, but it may be necessary

or desireable to set some positive value as a lower bound depending on the results of a given run.

The choice of upper bounds is more difficult. If they are set too high the feasible region in which BOXPLX searches for a minimum becomes large with a consequent heavy usage of computer time in reaching a minimum. If the bounds are too low, the "best" answer for a given run may be right on the bounds, necessitating a relaxation and further runs to proceed toward the minimum cost function. Some judgment is necessary in choosing these upper and lower bounds.

The choice of the starting values for the variables is no less critical in avoiding wastage of computer time. A preliminary root-locus investigation of the system may give some idea of starting values which will be close to the "best" values, but it was decided that the real power of the method described herein would be in its application to the case where one had no idea where the answers were. Thus, the criteria mentioned above were not brought to bear too heavily on the problem.

3. The Function and Constraint Evaluation Sub-Functions

The function evaluation sub-function, FE(X,XDATA), is the routine which actually calculates the cost-function. For every set (vertex) of poles and zeros generated by BOXPLX this routine simulates the system response comparing it at every time step with the desired response, subtracting the

difference, squaring it and integrating the result from $t = 0.0$ to $t = t_f$.

KE(X) checks to see if there are any implicit constraint violations in the generated vertex. Implicit constraints may be any arbitrary function of the variables the user may desire, e.g. the product of two poles must be less than a given number. In this investigation no implicit constraints were used.

4. The Desired Response (XDATA)

The response curve against which the system, with its current set of poles and zeros, is compared may be read into the programme as a set of data cards defining the amplitude of any arbitrary response at each time step of the simulation interval. Thus a curve which has no deterministic equation may be used to define a desired response.

If the equation of a desired response curve is known, this equation could be included in function FE and the amplitudes computed simultaneously with the actual system response.

In this investigation, data cards of the desired response were used, as being the most general method of specifying the response.

5. Subroutine Graph

This subroutine is appended to the main programme to plot the desired and actual response on a single graph when exit occurs from BOXPLX with the "best" set of poles and zeros. In addition, it can be made to plot other responses

which are not being compared with a desired response, to check on their quality.

C. CONVERSION OF SYSTEM EQUATIONS TO STATE VARIABLE FORM

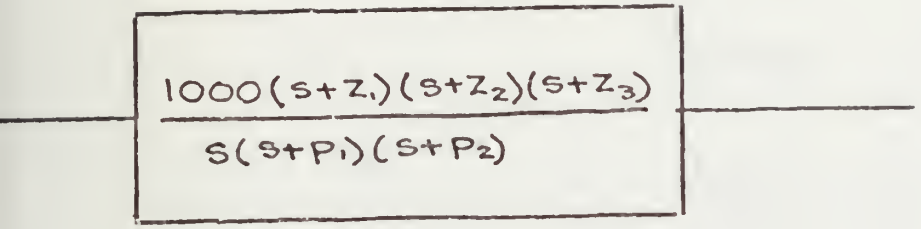
1. General

The system had previously been modeled by Huang using DSL (Digital Simulation Language) for which a program deck was available. However, since DSL is a language on its own, it was not possible to use it in its existing form; in an interactive mode in a Fortran program with the function minimization sub-routine. Thus, it was decided to first convert the system equations to state variable form so that they could be integrated by a built-in integration routine.

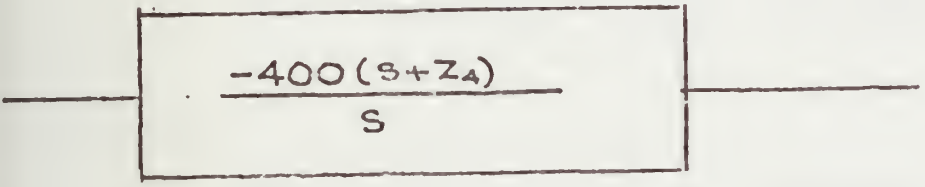
Since the poles and zeros of the compensators will be the unknown parameters and it is intended to use the same quantity and configuration of these parameters as in Huang's thesis, there are nine unknowns; three poles and six zeros. The configuration of the compensators with these unknowns is shown in Figure III-1, where the p_i are the poles and the z_j are the zeros.

2. The Plant Equations:

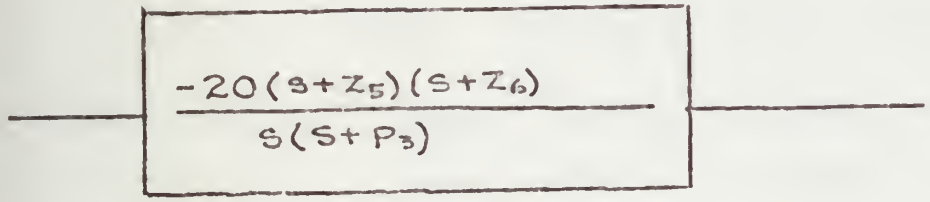
The plant equations in Figure II-2 as excerpted from Reference 4 are almost in state variable form already except for the $\dot{\alpha}$ term in equation (3). By substituting equation (2) into equation (3) and collecting terms the following state equations result:



Thrust Compensator



Flaps Compensator



Elevator Compensator

FIGURE III - 1

$$\dot{u} = X_u u \quad + X_\alpha \alpha \quad - g\theta$$

$$\dot{\alpha} = \frac{Z_u u}{V_o(1-Z_{\dot{\alpha}})} + \frac{Z_w \alpha}{1-Z_{\dot{\alpha}}} + \left[\frac{1 + \frac{Z_q}{V_o}}{1 - Z_{\dot{\alpha}}}\right] q + \frac{Z_{\delta_t} \cdot \delta t}{V_o(1-Z_{\dot{\alpha}})}$$

$$+ \frac{Z_{\delta_f} \cdot \delta_f}{V_o(1-Z_{\dot{\alpha}})} + \frac{Z_{\delta_e} \cdot \delta_e}{V_o(1-Z_{\dot{\alpha}})}$$

$$\dot{q} = \left[M_u + \frac{M_{\dot{\alpha}} Z_u}{V_o(1-Z_{\dot{\alpha}})}\right] u + \left[M_\alpha + \frac{M_{\dot{\alpha}} Z_w}{1-Z_{\dot{\alpha}}}\right] \alpha + \left[M_q + \frac{M_{\dot{\alpha}}(1-\frac{Z_q}{V_o})}{1 - Z_{\dot{\alpha}}}\right] q$$

$$+ \left[M_{\delta_t} + \frac{M_{\dot{\alpha}} Z_{\delta_t}}{V_o(1-Z_{\dot{\alpha}})}\right] \delta_t + \left[M_{\delta_f} + \frac{M_{\dot{\alpha}} Z_{\delta_f}}{V_o(1-Z_{\dot{\alpha}})}\right] \delta_f + \left[M_{\delta_e} \frac{M_{\dot{\alpha}} Z_{\delta_e}}{V_o(1-Z_{\dot{\alpha}})}\right] \delta_e$$

$$\dot{\theta} = q$$

Substitution of the force and moment coefficients from Reference 4 into these equations results in the following plant state equations:

$$\dot{u} = -.0670u + 22.2\alpha \quad -32.2\theta + .0670\delta_t - 6.12\delta_f \tag{5}$$

$$\dot{\alpha} = -.004017u - .7623\alpha + .9479q - .001355\delta_t - .1527\delta_f - .0752\delta_e \tag{6}$$

$$\dot{q} = .003785u - 1.826\alpha - 1.452q \quad -.001732\delta_t + .09953\delta_f - 1.9451\delta_e \tag{7}$$

$$\dot{\theta} = q \tag{8}$$

3. The Actuator Equations

a; Thrust actuator: The thrust actuator, in the notation of Figure II-2, has y_1, as input and δ_t as output with transfer function:

$$\frac{1.21}{s^2 + 1.98s + 1.21}$$

In differential equation form this becomes:

$$\ddot{\delta}_t + 1.98\dot{\delta}_t + 1.21\delta_t = 1.21\ y_1$$

If we let $\dot{\delta}_t = \delta_{td}$ the state equations for the thrust actuator become:

$$\dot{\delta}_t = \delta_{td} \tag{9}$$

$$\dot{\delta}_{td} = -1.21\delta_t - 1.98\delta_{td} + 1.21\ y_1 \tag{10}$$

b. Flaps actuator: The flaps actuator has input y_2 and δ_f as output with transfer function:

$$\frac{1.0}{s + 1.0}$$

In differential equation form this becomes:

$$\dot{\delta}_f + \delta_f = y_2$$

Thus the state equation for the flaps actuator is:

$$\dot{\delta}_f = -\delta_f + y_2 \tag{11}$$

c. Elevator actuator: The elevator actuator has input y_3 and output δ_e with transfer function:

$$\frac{121.0}{s^2 + 6.6s + 121.0}$$

The differential equation for this transfer function is:

$$\ddot{\delta}_e + 6.6\ \dot{\delta}_e + 121.0\ \delta_e = 121.0 y_3$$

Letting $\dot{\delta}_e = \delta_{ed}$ the state equations are:

$$\dot{\delta}_e = \delta_{ed} \tag{12}$$

$$\dot{\delta}_{ed} = -6.6\,\delta_{ed} - 121.0\,\delta_e + 121.0\,y_3 \qquad (13)$$

4. The Compensator Equations:

In the notation which follows, the ij subscripts are defined as follows: the i subscript refers to the thrust, flaps and elevator compensators as 1,2,3 respectively. The j subscript is the index of the coefficients in each compensator. The method of transforming the transfer functions into state equations when the resulting differential equation results in derivatives of the output is as set out in reference 5. In the notation of that reference, given an n^{th} order differential equation:

$$x^{(n)}(t) + a_1 x^{(n-1)}(t) + \ldots\ldots a_{n-1} x^{(1)}(t) + a_n x(t) =$$

$$b_o u^{(n)}(t) + b_1 u^{(n-1)}(t) + \ldots\ldots b_{n-1} u^{(1)}(t)$$

$$+ b_n(t)$$

where $x^{(n)}(t) = \dfrac{d^n x(t)}{dt^n}$

A set of state variables for this equation can be defined as:

$$x_1(t) = x(t) - \beta_o u(t)$$

$$x_2(t) = \dot{x}_1(t) - \beta_1 u(t)$$

$$\vdots$$

$$x_n(t) = \dot{x}_{n-1}(t) - \beta_{n-1} u(t)$$

Where

$$\beta_o = b_o$$
$$\beta_1 = b_1 - a_1\beta_o$$
$$\beta_2 = b_2 - a_1\beta_1 - a_2\beta_2$$
$$\vdots$$
$$\beta_n = b_n - a_1\beta_{n-1} - \cdots\cdots - a_{n-1}\beta_1 - a_n\beta_o$$

With this choice of state variable, the state equations are:

$$\dot{x}_1(t) = x_2(t) + \beta_1 u(t)$$
$$\dot{x}_2(t) = x_3(t) + \beta_2 u(t)$$
$$\vdots$$
$$\dot{x}_{n-1}(t) = x_n(t) + \beta_{n-1} u(t)$$
$$\dot{x}_n(t) = -a_n x_1(t) - a_{n-1} x_2(t) - a_{n-2} x_3(t) \cdots\cdots - a_n x_n(t) + \beta_n u(t)$$

a. Thrust compensator: The thrust compensator has input e_1, output y_1 and the transfer function is:

$$\frac{y_1}{e_1} = \frac{1000(s+z_1)(s+z_2)(s+z_3)}{s(s+p_1)(s+p_2)}$$

Multiplying out and transforming to a differential equation we have:

$$\dddot{y} + (p_1+p_2)\ddot{y}_1 + p_1p_2\dot{y} = 1000\,\ddot{e}_1 + (z_1+z_2+z_3)\ddot{e}_1 + (z_1z_2+z_2z_3+z_1z_3)\dot{e}_1 + (z_1z_2z_3)e_1$$

Let: $a_{11} = p_1+p_2$, $a_{12} = p_1p_2$, $a_{13} = 0$

$b_{10} = 1000$, $b_{11} = 1000(z_1+z_2+z_3)$, $b_{12} = 1000(z_1z_2+z_2z_3+z_1z_3)$

$b_{13} = 1000z_1z_2z_3$

Thus the state variables are:

$$w_1 = y_1 - \beta_{10}e_1 \qquad w_2 = \dot{y}_1 - \beta_{11}e_1 \qquad w_3 = \ddot{y}_1 - \beta_{12}e_1$$

Where

$$\beta_{10} = b_{10} = 1000, \quad \beta_{11} = b_{11} - \beta_{10}a_{11}, \quad \beta_{12} = b_{12} - \beta_{11}a_{11} - \beta_{10}a_{12},$$

$$\beta_{13} = b_{13} - \beta_{12}a_{11} - \beta_{11}a_{12}$$

and the state equations are:

$$\dot{w}_1 = w_2 + \beta_{11}e_1 \qquad (14)$$

$$\dot{w}_2 = w_3 + \beta_{12}e_1 \qquad (15)$$

$$\dot{w}_3 = -a_{12}w_2 - a_{13}w_3 + \beta_{13}e_1 \qquad (16)$$

The required output y_1, when these equations are solved will then be: $y_1 = w_1 + \beta_{10}e_1$

b. Flaps compensator: The flaps compensator has input e_2, output y_2 and its transfer function is:

$$\frac{y_2}{e_2} = \frac{-400(s+z_4)}{s}$$

Multiplying out and transforming to a differential equation we have:

$$\dot{y}_2 = -400\dot{e}_2 - 400z_4e_2$$

Let $a_{21} = 0$

$$b_{20} = -400 \ , \ b_{21} = -400z_4$$

Thus for this first order subsystem, the state variable is:

$$w_4 = y_2 - \beta_{20}e_2$$

Where $\beta_{20} = b_{20} = -400$, $\beta_{21} = b_{21} - \beta_{20}a_{21}$

and the state equation is:

$$\dot{w}_4 = \beta_{21} e_2 \qquad (17)$$

The required output y_2 on solution of this equation will be:

$$y_2 = w_4 + \beta_{20} e_2$$

c. Elevator Compensator:

The input to the elevator compensator is e_3 and its output is y_3 and it has transfer function:

$$\frac{y_3}{e_3} = \frac{-20(s+z_5)(s+z_6)}{s(s+p_3)}$$

Multiplying out and transforming to a differential equation we have:

$$\dddot{y}_3 + p_3 \ddot{y}_3 = -20\,\ddot{e}_3 + (z_5+z_6)\dot{e}_3 + z_5 z_6 e_3$$

Let $a_{31} = p_3$, $a_{32} = 0$

$b_{30} = -20$, $b_{31} = -20(z_5+z_6)$, $b_{32} = -20 z_5 z_6$

The state variables are:

$$w_5 = y_3 - \beta_{30} e_3 \qquad w_6 = \dot{y}_3 - \beta_{31} e_3$$

Where $\beta_{30} = b_{30} = -20$, $\beta_{31} = b_{31} - \beta_{30} a_{31}$, $\beta_{32} = b_{32} - \beta_{31} a_{31} - \beta_{30} a_{32}$

and the state equations are

$$\dot{w}_5 = w_6 + \beta_{31} e_3 \qquad (18)$$

$$\dot{w}_6 = -a_{31} w_6 + \beta_{32} e_3 \qquad (19)$$

On solution of these equations the required output y_3 is obtained from:

$$y_3 = w_5 + \beta_{30} e_3$$

5. The Feedback and Cost Function Equations

The feedback loops shown in Figure II-2 are included by simply adding three more equations rather than substituting in the state equations. This allows for more ease and flexibility in computer programming of the system model. The three equations are:

$$e_1 = r_1 - u$$

$$e_2 = r_2 - \alpha$$

$$e_3 = r_3 - \theta$$

The general form of the cost function J discussed in Section above as applied to this system is:

$$J = \int_o^{t_f} (u_{ref}-u)^2 + (\alpha_{ref}-\alpha)^2 + (\theta_{ref}-\theta)^2 \; dt$$

6. The Complete System State Equations:

Changing the variables in equations 5 to 19 inclusive to a consistent set of variables as shown in Table III-1 the following complete system equations result:

$$\dot{x}_1 = -.067x_1+22.2x_2-32.2x_4-6.12x_7+.067x_5$$

$$\dot{x}_2 = -.004017x_1-.7263x_2+.9479x_3-.001355x_5-.1527x_7-.0752x_8$$

$$\dot{x}_3 = .003755x_1-1.826x_2-1.452x_3-.001732x_5+.09953x_7-1.945x_8$$

$$\dot{x}_4 = x_3$$

$$\dot{x}_5 = x_6$$

$$\dot{x}_6 = -1.21x_5-1.98x_6+1.21x_{10}+1210.0e_1$$

$$\dot{x}_7 = -x_7+x_{13}-400.0e_2$$

$$\dot{x}_8 = x_9$$

$$\dot{x}_9 = -121.0x_8-6.6x_9+121.0x_{14}-2420.0e_3$$

$$\dot{x}_{10} = x_{11}+\beta_{11}e_1$$

$$\dot{x}_{11} = x_{12}+\beta_{12}e_1$$

$$\dot{x}_{12} = -a_{12}x_{11}-a_{11}x_{12}+\beta_{13}e_1$$

$$\dot{x}_{13} = \beta_{21}e_2$$

$$\dot{x}_{14} = x_{15}+\beta_{31}e_3$$

$$\dot{x}_{15} = -a_{31}x_{15}\beta_{32}e_3$$

$$e_1 = r_1-x_1$$

$$e_2 = r_2-x_2$$

$$e_3 = r_3-x_4$$

These equations and the defining equations for the coefficients in terms of the free parameters (the poles and zeros) are coded in Fortran and appear in the sample program appended. The equations appear both in the function evaluation subfunction to BOXPLX for calculating J and in subroutine GRAPH for plotting the results. Further computer symbols used in this programme are explained in Table III-1. The computational flow chart showing the inter-relationships in the total programme is shown in Figure III-2.

TABLE III - 1

Variable Symbols

Variable	State Variable	Computer Symbol
u	x_1	X(1)
$\dot{u}$	$\dot{x}_1$	XDOT(1)
$\dot{\alpha}$	x_2	X(2)
α	$\dot{x}_2$	XDOT(2)
q	x_3	X(3)
$\dot{q}$	$\dot{x}_3$	XDOT(3)
θ	x_4	X(4)
$\dot{\theta}$	$\dot{x}_4$	XDOT(4)
δ_t	x_5	X(5)
$\dot{\delta}_t, \delta_{td}$	x_6	X(6)
$\ddot{\delta}_t$	$\dot{x}_6$	XDOT(6)
δ_f	x_7	X(7)
$\dot{\delta}_f$	$\dot{x}_7$	XDOT(7)
δ_e	x_8	X(8)
$\dot{\delta}_e, \delta_{ed}$	x_9	X(9)
w_1	x_{10}	X(10)
w_2	x_{11}	X(11)
w_3	x_{12}	X(12)
w_4	x_{13}	X(13)
w_5	x_{14}	X(14)
w_6	x_{15}	X(15)
J		X(16)
p_1		PA(1)
p_2		PA(2)
p_3		PA(3)
z_1		PA(4)
z_2		PA(5)
z_3		PA(6)
z_4		PA(7)
z_5		PA(8)

TABLE III - 1 (cont.)

Variable	Computer Symbol
z_6	PA(9)
e_1	E1
e_2	E2
e_3	E3
r_1	R1
r_2	R2
r_3	R3
Upper bounds	BU(I)
Lower bounds	BL(I)
Starting values	XS(I)
Desired system output	XDATA(I), XDATAJ(I)
Intermediate functions of parameters	AIJ, BIJ, BETAIJ
Time	T
Time step	DT

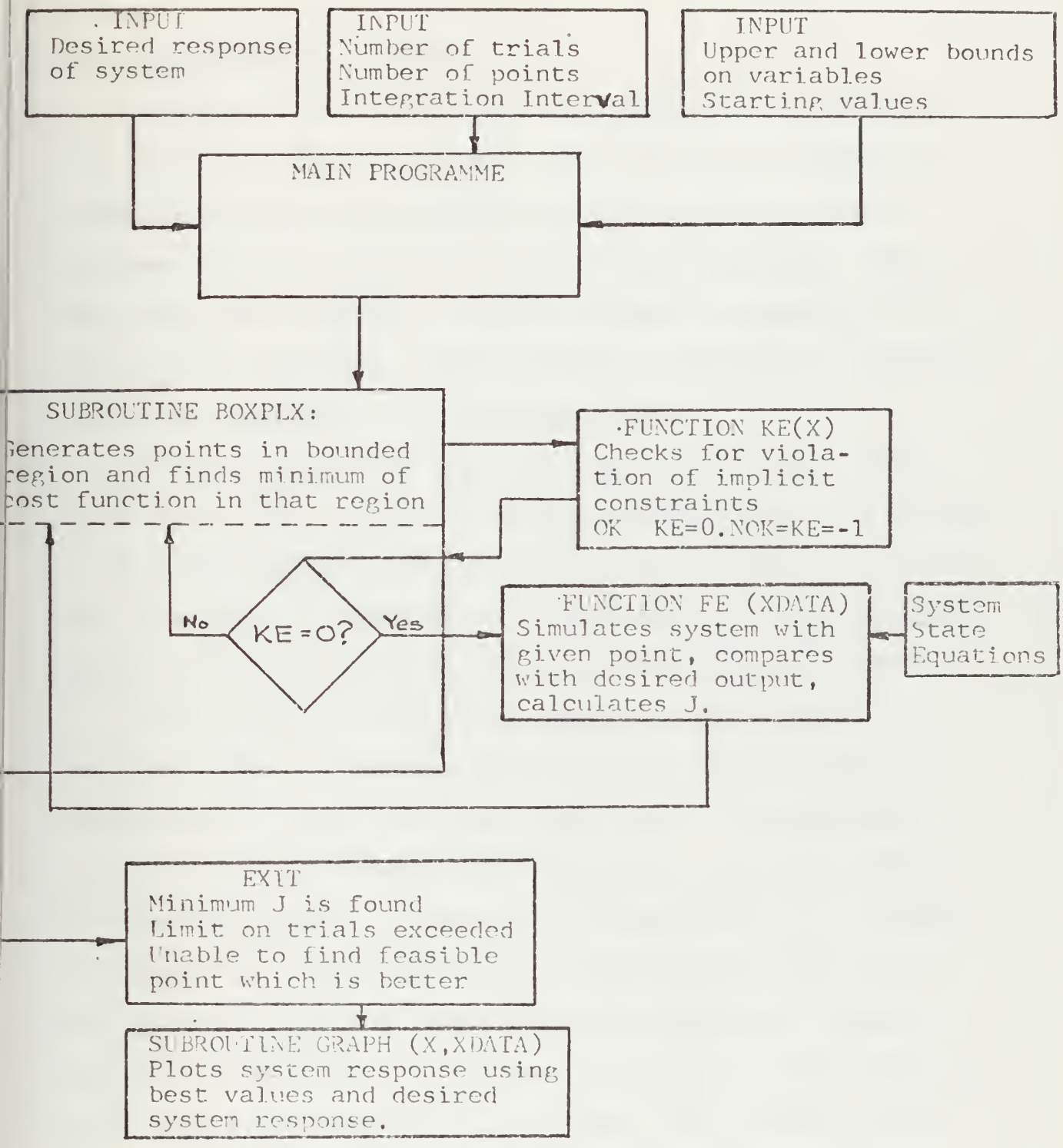


FIGURE III-2 Computational Flow Chart

D. INVESTIGATIVE APPROACH

Since it was not known how well BOXPLX would perform in this task of finding the poles and zeros it was decided to do all the initial investigation with the known original response with the poles and zeros as found by Huang. When this phase was completed, it was intended to specify an arbitrary but reasonable output response and attempt a matching of system response to this arbitrary one.

As was noted earlier the only input used for all phases was a step input $r_3 = -0.25$ radians corresponding to a change in the pitch angle $\theta = -0.25$ radians. Thus there are two outputs, u and α which are not commanded by a corresponding input; and the question arises as to which output to make J a function of. This would depend on the situation in the real world. If the pitch angle were important due to a requirement to stay on a glide slope, then J should be a function of θ. If speed were important due to the distance to touchdown, then J should be a function of u. Ideally J should be a function of all three outputs, but it may not be easy to specify just what they should be. The approach used in the initial phase was to let J be a function of an output not commanded by an input (i.e., speed), then let J be a function of pitch angle and finally a function of all three outputs. It should be remembered that in this initial phase, all three responses, and the poles and zeros which produced them, are known. The only other question to

be resolved is that of the starting values and upper and lower bounds for the variables. Until the capabilities of BOXPLX had been checked it was decided to offset the starting values approximately 25% from the known values (the so-called standard start point) with arbitrary close upper and lower bounds. These are shown in Table III - 2 together with the actual values.

	Pole 1	Pole 2	Pole 3
Actual	4.0	10.0	10.0
XS	3.5	11.0	9.0
BU	5.0	12.0	12.00
BL	3.0	8.0	8.00

	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Actual	1.0	0.5	0.5	0.5	0.3	0.4
XS	0.75	0.55	0.35	0.55	0.25	0.45
BU	1.5	0.7	0.7	0.7	0.4	0.5
BL	0.5	0.3	0.3	0.3	0.2	0.3

Standard Starting Values

TABLE III - 2

If BOXPLX proved capable of locating the correct poles and zeros to a desired degree of precision, the next step was to have it try with the so-called arbitrary start point. For this case it will be assumed that the pole and zero locations are unknown and the starting values will all be set at XS = 10.0, the upper bounds at BU = 20.0 and the lower bound at BL = 0.0.

Assuming success in this initial phase it was planned to then specify an arbitrary pitch angle response using Whiteley's standard form of response to a step input with zero position

error and 10% maximum overshoot with an appropriate ω_o (Ref. 7). The speed response would then be checked and if "satisfactory", success will have been achieved. If not, a speed response using an overdamped second order impulse response would be combined with the pitch angle response to make a combined cost function.

V. RESULTS AND COMMENTS

A. INITIAL PHASE

The initial phase, it will be recalled, consisted of investigating the capabilities of BOXPLX when the poles and zeros of the desired response were known, i.e., the original poles and zeros as determined by Huang. It was also during this phase that various methods were used to reduce the amount of computer time involved in a run and to get an idea of how large the value of the cost function could be and still have an acceptably close response match.

Table IV - 1 shows the results of various runs made with J a function of speed (i.e., a non-commanded output) and starting with the standard start point. The value of the cost function varies from $J = .00036$ to $J = .052$. The response curves for the smallest value are so superimposed as to be indistinguishable and those for the largest value are shown in Figures IV - 1, IV - 2 and IV - 3 for u, and α and θ respectively. Although the largest cost function is two orders of magnitude larger than the smallest one, the response matching is quite acceptable. Thus it appears that it is not necessary for BOXPLX to reach a minimum of the cost function to give good results and recognition of this fact could cut the computer time dramatically.

The big differences in Table IV - 1 are in the computer time involved with each run. It was during this initial phase that it was decided to halve the simulation time to 12.5

RESULTS OF VARIOUS BOXPLX RUNS FOR SPEED

Standard Starting Point

	J	POLE 1	POLE 2	POLE 3	ZERO 1	ZERO 2	ZERO 3	ZERO 4	ZERO 5	ZERO 6	TRIALS	TIME
Short Sim.	.0003655	3.999	9.995	9.934	1.011	0.471	0.517	0.464	0.269	0.420	873	24Min
Long Sim.	.0018269	3.993	9.996	9.951	0.975	0.467	0.549	0.470	0.311	0.392	1217	90Min
Short Sim. 9 Vertices	.00225	4.016	9.996	9.985	1.022	0.516	0.467	0.484	0.217	0.497	395	10Min
Short Sim.	.00546	3.937	10.040	0.650	0.990	0.523	0.462	0.341	0.215	0.436	600	17Min
Long Sim.	.004015	3.995	9.988	9.888	0.992	0.479	0.519	0.433	0.294	0.402	911	60Min
Long Sim.	.013393	4.089	9.867	9.850	1.940	0.480	0.504	0.411	0.288	0.411	368	30Min
Short Sim.	.05293	3.901	10.137	9.639	1.045	0.510	0.427	0.327	0.194	0.439	300	8Min
Actual		4.00	10.00	10.00	1.00	0.50	0.50	0.50	0.30	0.40		

TABLE IV - 1

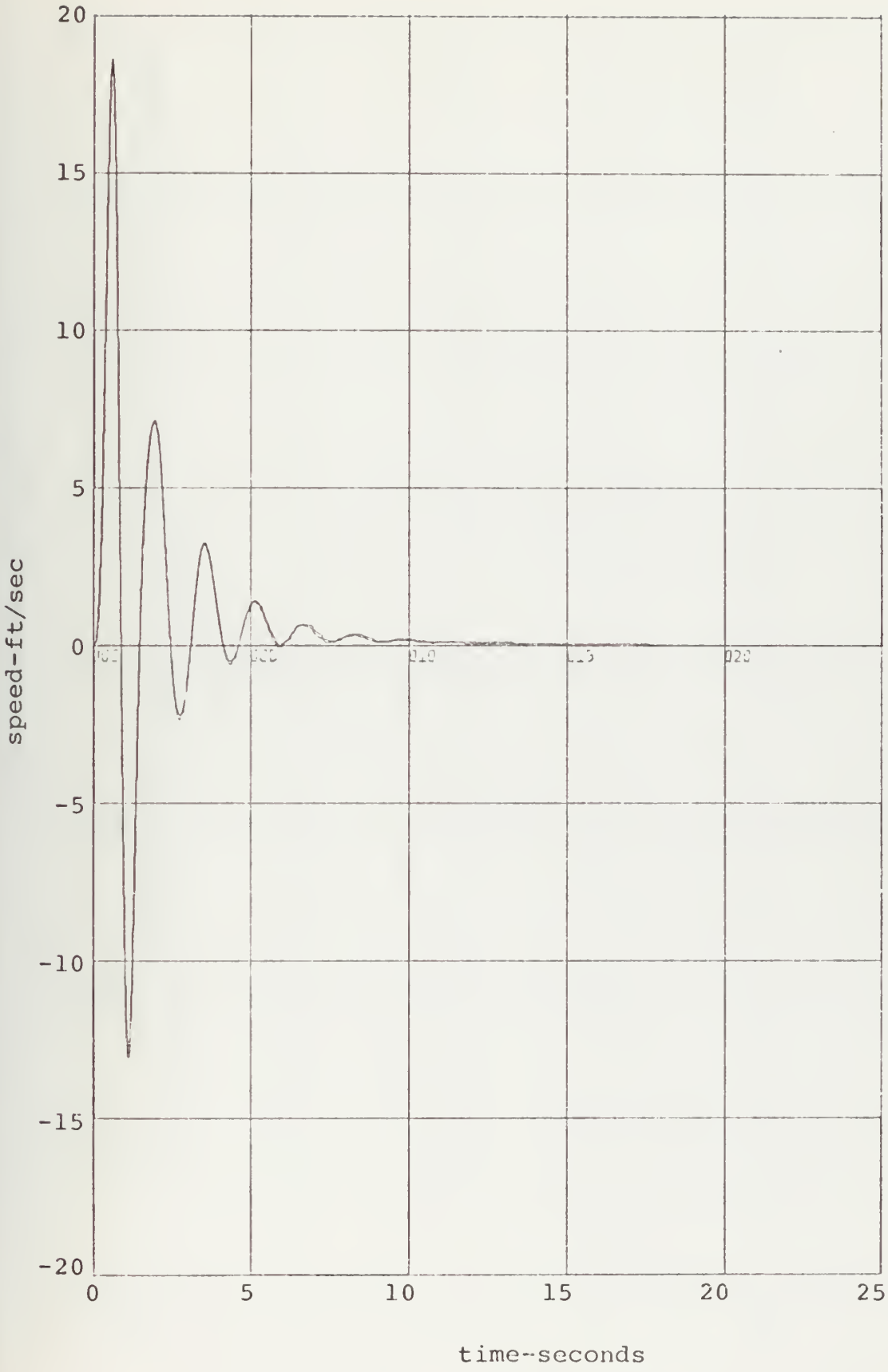


Figure IV - 1

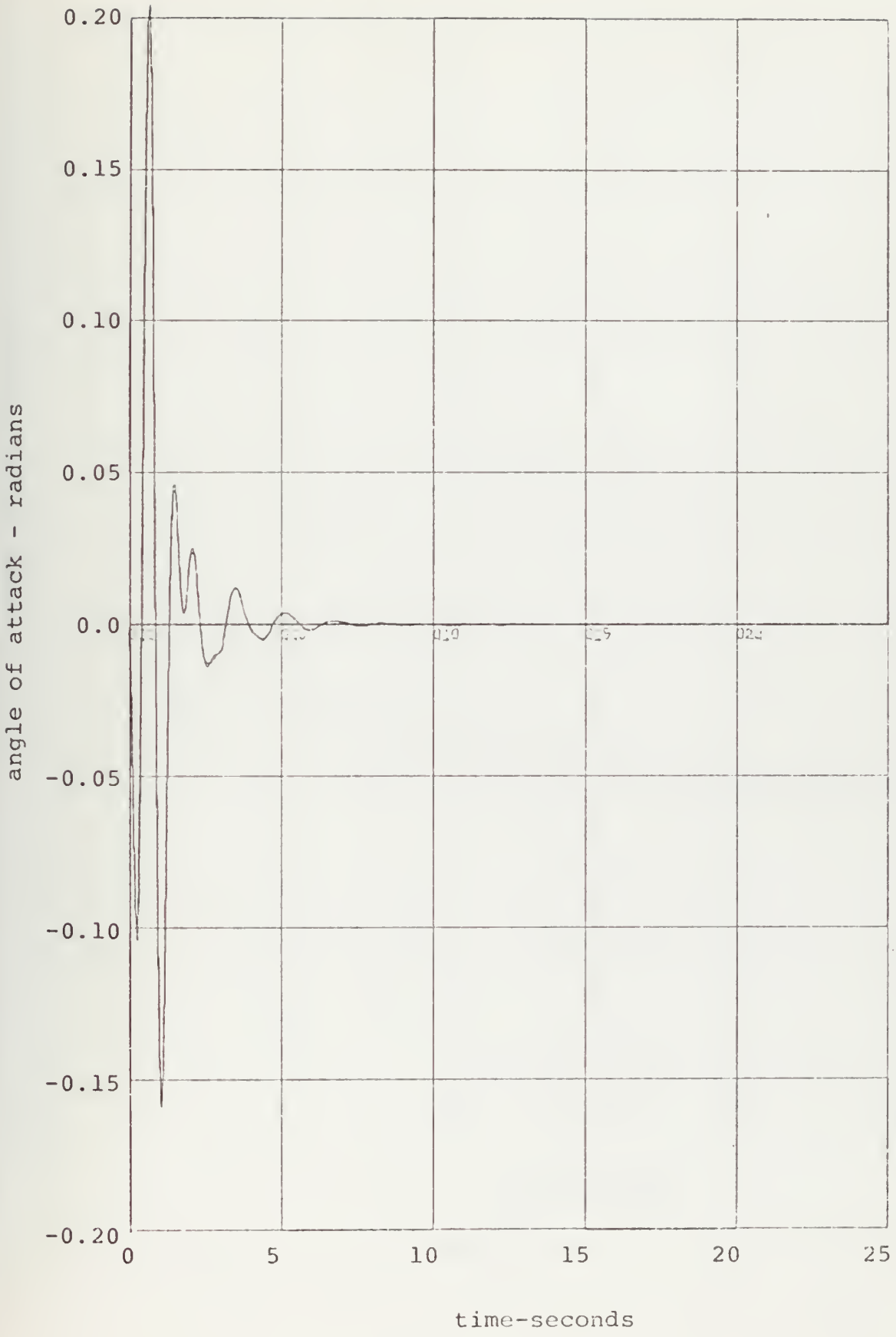

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 2

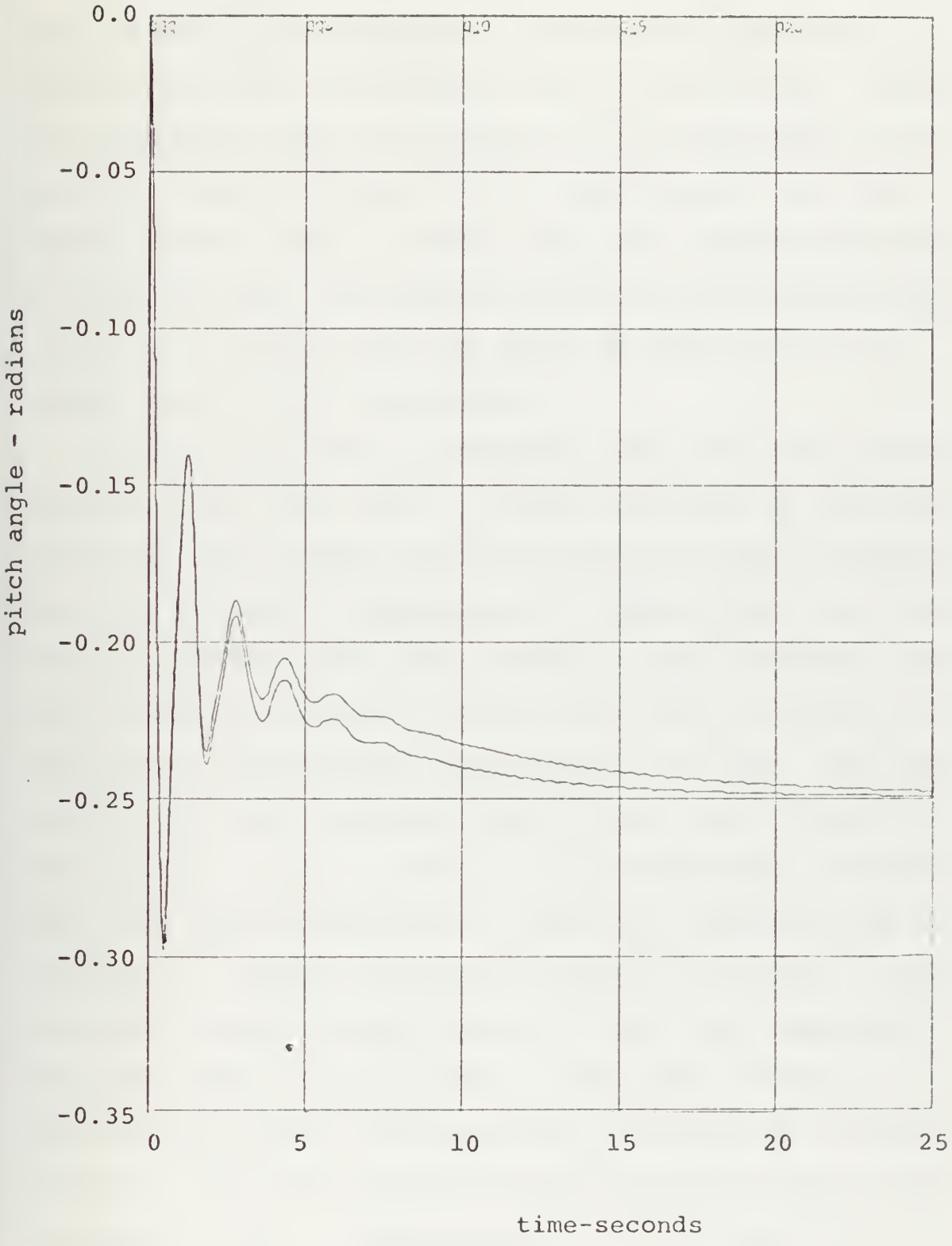

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 3

seconds for the calculations in subfunction FE(X) and the results show that the computer time is also halved. Another ploy attempted was to have BOXPLX set up each complex with only n vertices instead of 2n vertices and this again halved computer time. However, this last method was found to work only when the starting values of the variables were close to the minimum and thus would be used only in the latter stages of an investigation.

The next step was to investigate the effect of using the standard start point with J still a function of speed but enlarging the feasible region by setting the upper bounds at 20.0 and the lower bounds at 0.0. BOXPLX found a cost function of 0.05995 in 600 trials which is about twice the number of trials it took to find the last entry in Table IV - 1 which has a cost function approximately the same. The responses with poles and zeros found on this run are shown in Figures IV - 4, IV - 5 and IV - 6 for speed angle of attack and pitch angle respectively. Again the results are quite acceptable. Another interesting result of this run is that the pole and zero values differ, in some cases appreciably both from those obtained from the last run in Table IV - 1 and from the actual values, and yet the responses are quite similar. The values obtained from this run are shown below in Table IV - 2. A further mention of this sensitivity of the results will be made later.

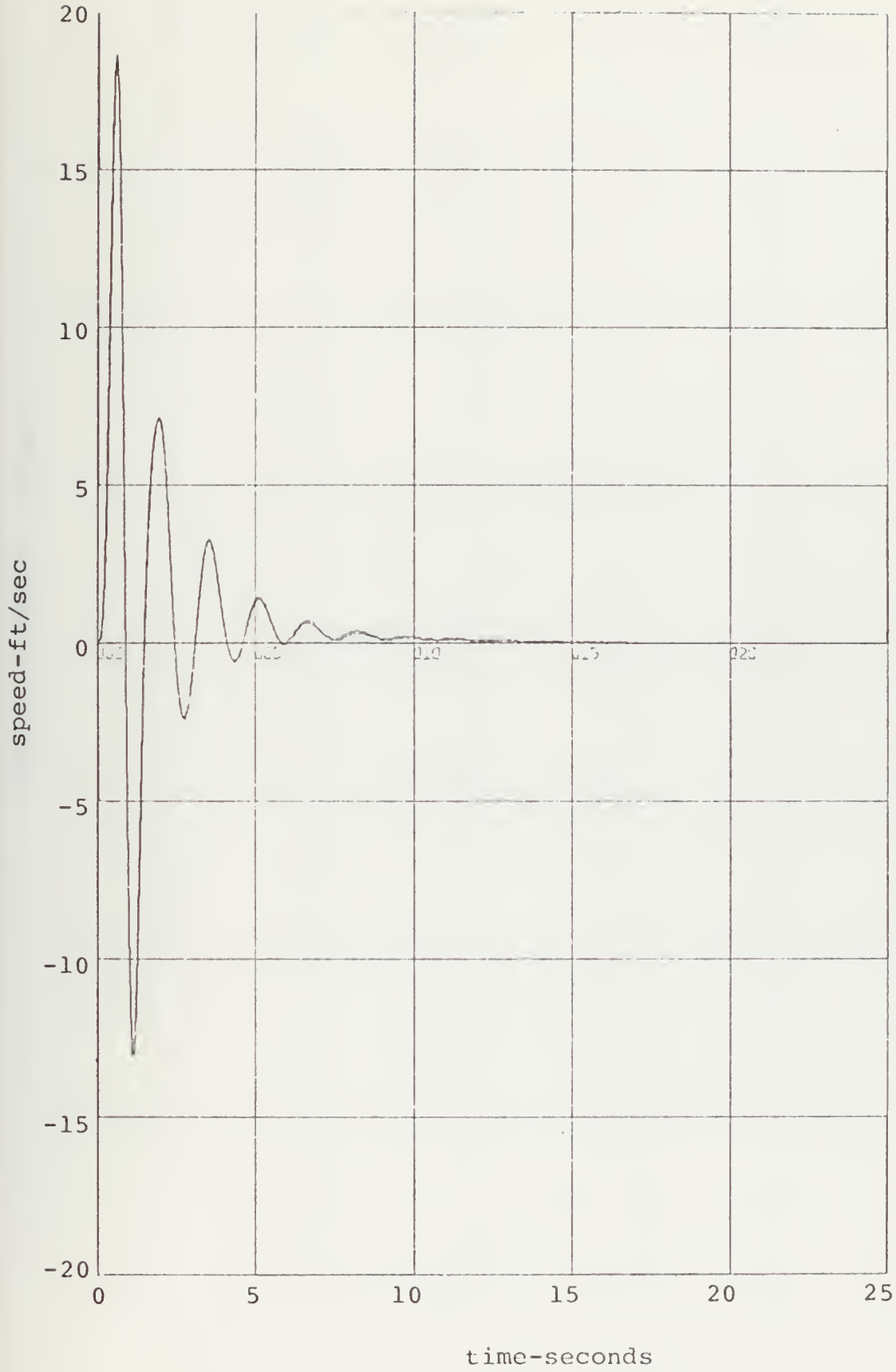


Figure IV - 4

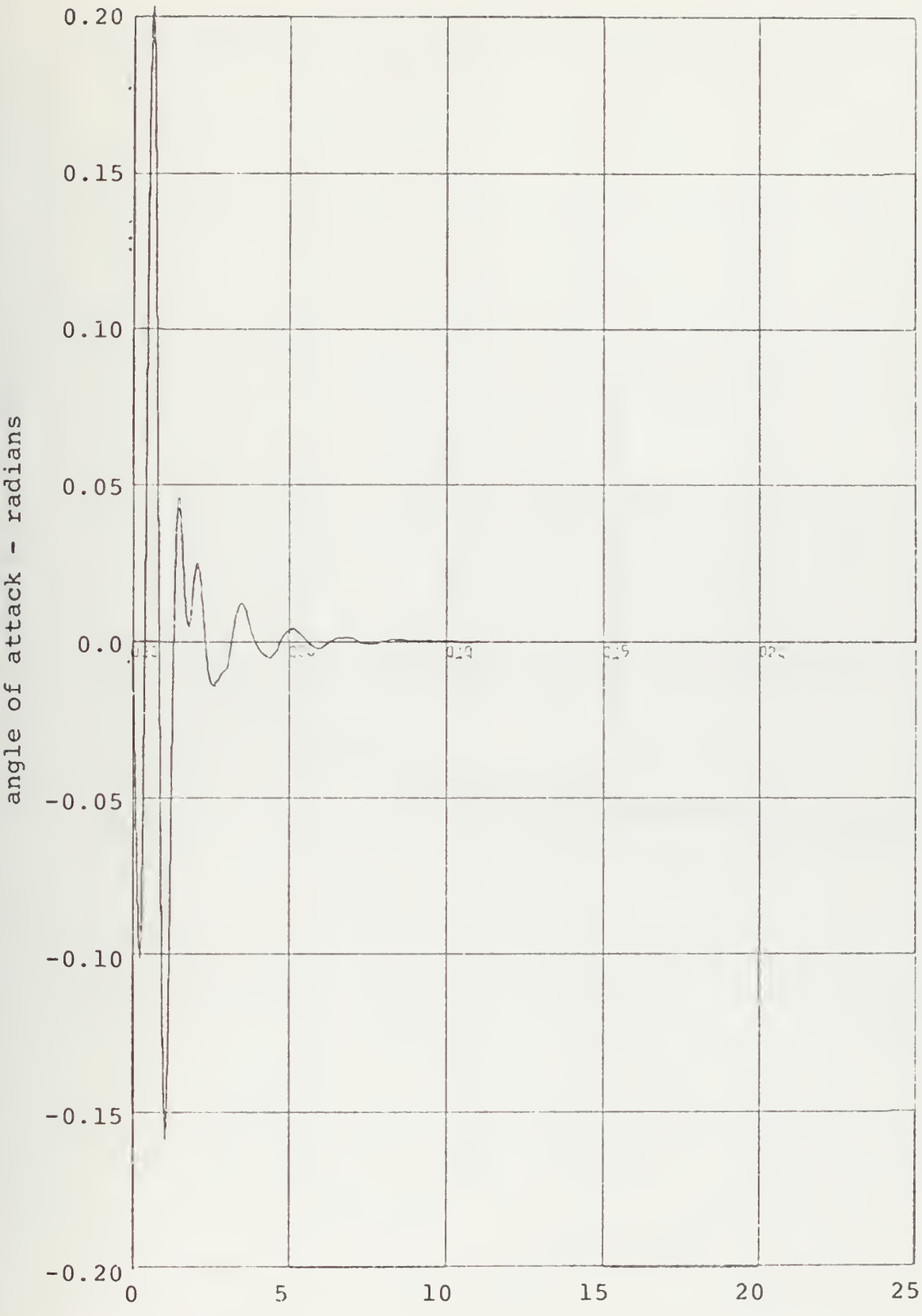

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 5

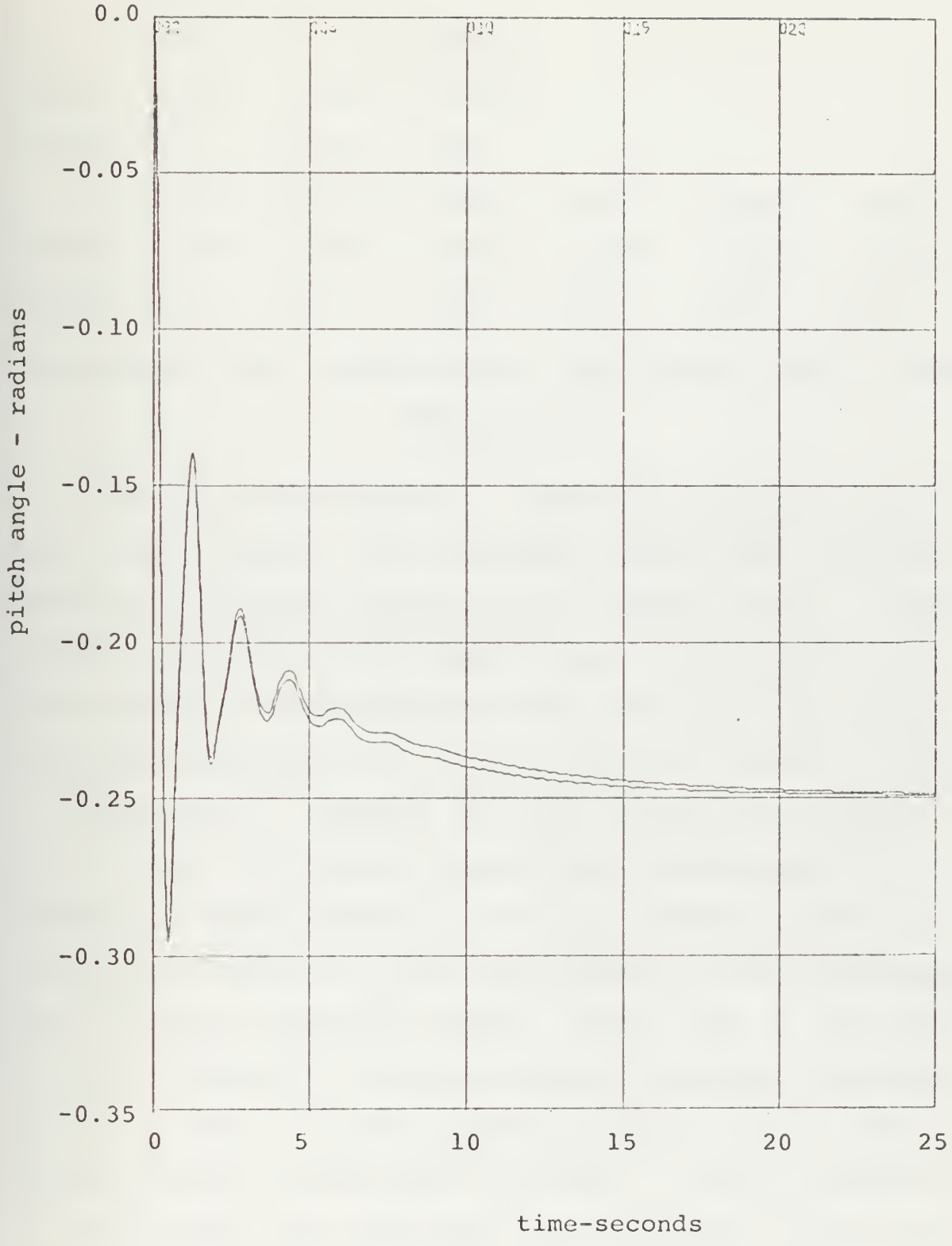

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 6

	Pole 1	Pole 2	Pole 3			
Found	4.409	0.437	10.013			
Actual	4.0	10.0	10.0			
	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Found	0.664	0.336	1.107	0.447	0.237	0.443
Actual	1.0	0.5	0.5	0.5	0.3	0.4

Parameters found, standard start, wide bounds, J=f(u) = .05995

Table IV - 2

Next, the cost function J was made a function of the pitch angle output, θ (the commanded ouput) again using the standard start point and the larger feasible region. In 600 trials, taking 17 minutes BOXPLX found a $J=6.8 \times 10^{-5}$ and the results of the simulation of the poles and zeros found are shown in Figures IV - 7, IV - 8 and IV - 9 for u , α and θ respectively. Although the cost function as a function of θ cannot be compared directly with the one which is a function of speed because of the scale factors involved it can be considered to a first approximation to be approximately three orders of magnitude smaller. Thus, this J as a function of θ would be roughly equivalent to one as a function of u of .068. But the important point to note is that the response match is much better than when J was a function of u and in fact the values found for the poles and zeros correspond to the actual ones with a maximum of 12% with another 8.8% and the rest less than 2%. Thus it would appear that

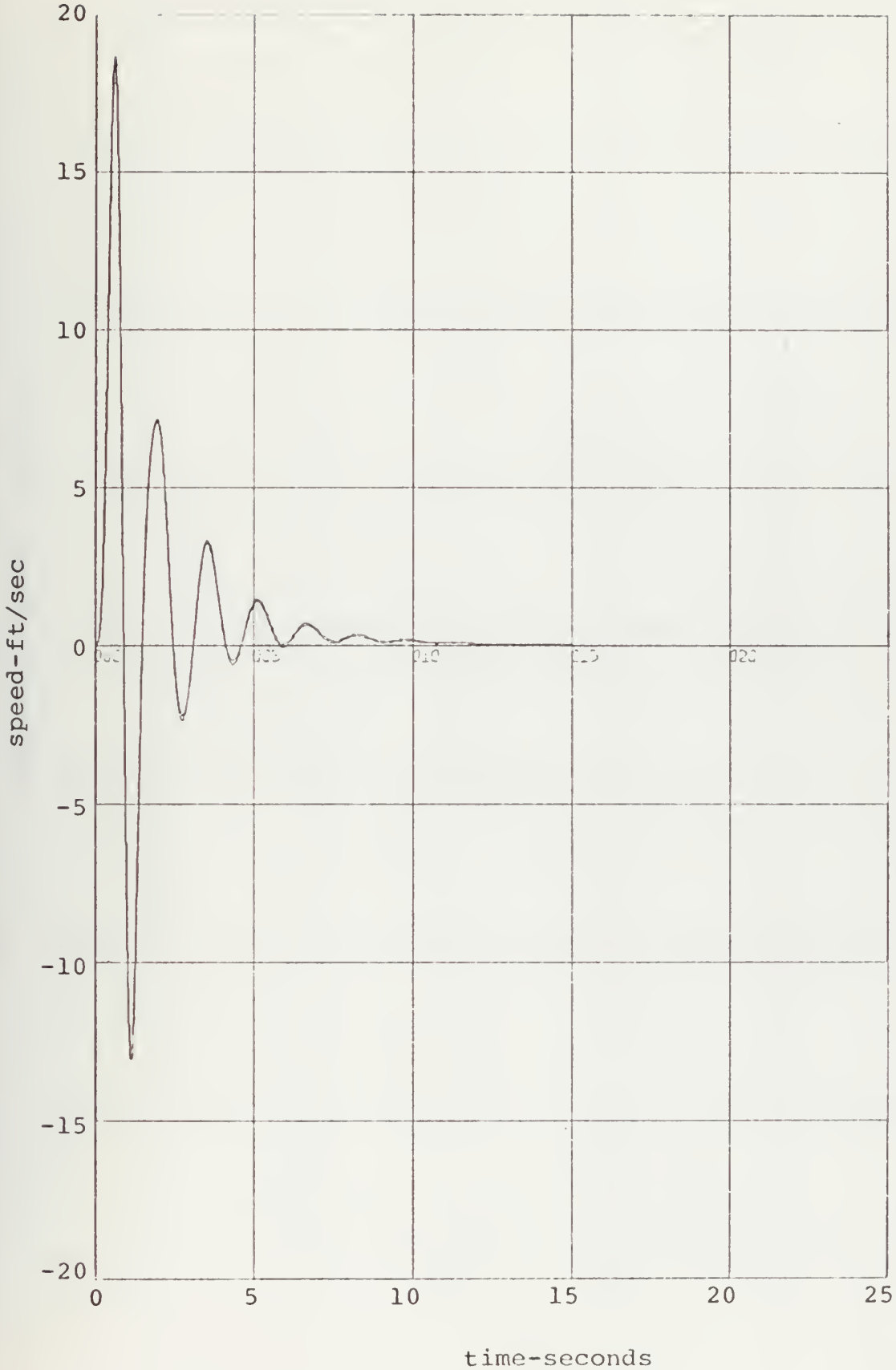


Figure IV - 7

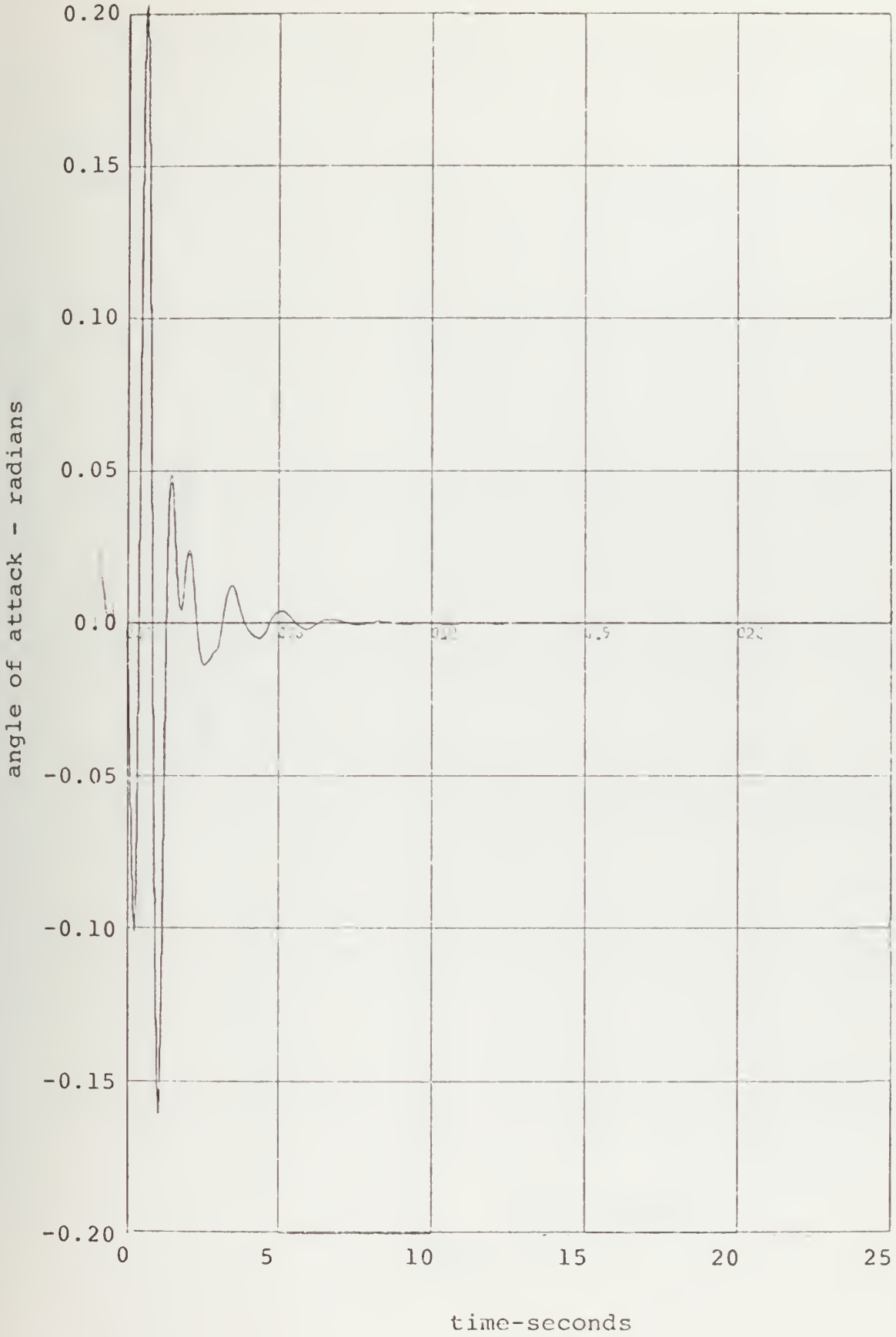

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 8

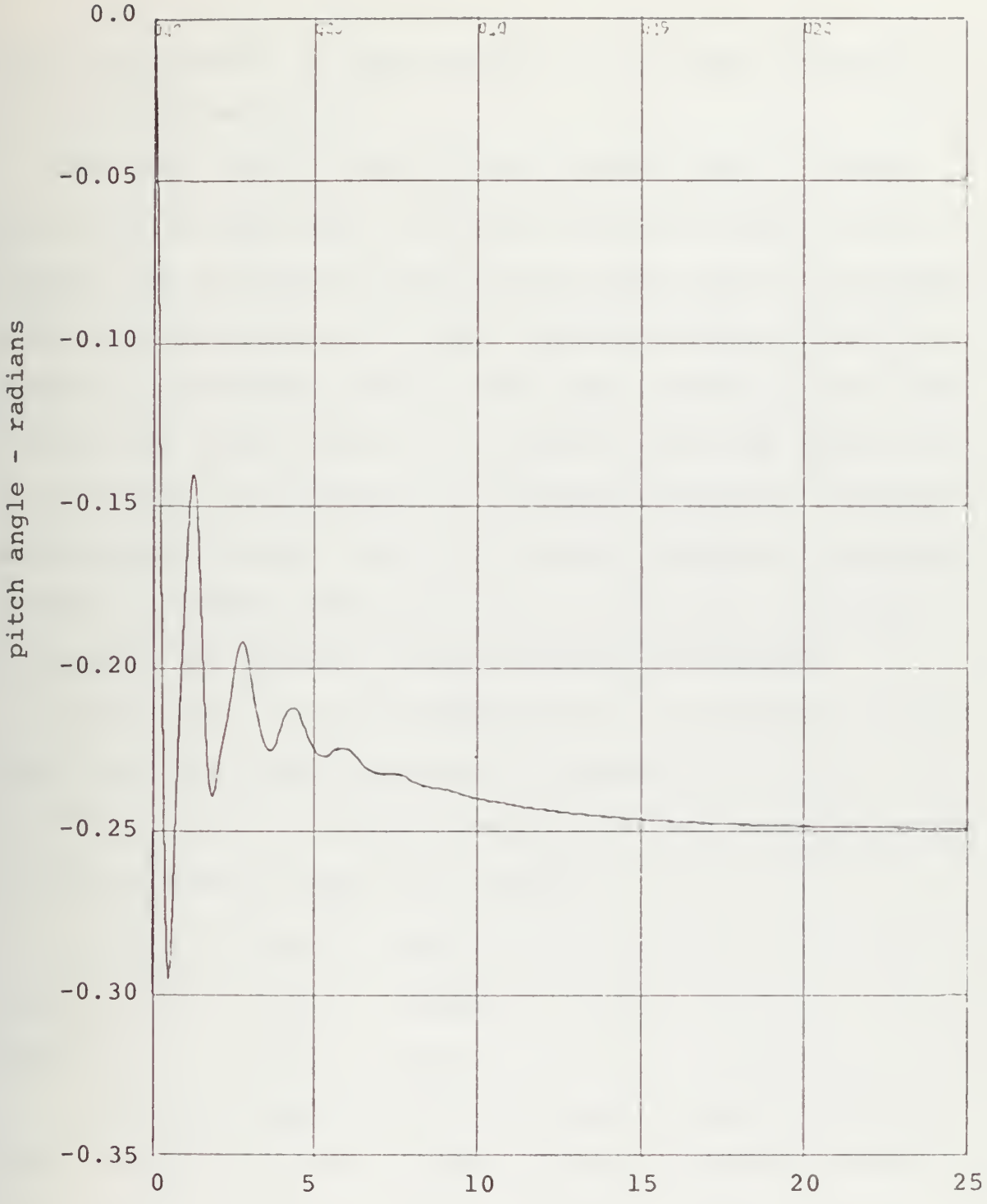

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 9

better convergence is obtained when J is made a function of the commanded output.

Next came the real test of the capabilities of BOXPLX. Here it was assumed that the actual poles and zeros of the response are unknown and BOXPLX would start from a so-called arbitrary start point in a large feasible region. For this purpose, as mentioned earlier, the upper bounds were all set at 20.0, the lower bounds at 0.0 and the starting values at 10.0, as might be assumed by a reasonably competent engineer, examining this model. The cost function was made a function of speed and then a function of the pitch angle, as before, to check on the finding in the preceeding paragraph.

First, with J = f(u), BOXPLX found J = 0.99016 in 736 trials with the responses shown in Figures IV - 10, IV - 11 and IV - 12 for u , α and θ respectively and found the poles and zeros shown in Table IV - 3 below.

	Pole 1	Pole 2	Pole 3			
Found	5.328	8.411	12.289			
Actual	4.0	10.0	10.0			
	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Found	0.229	0.722	1.342	1.317	0.084	1.130
Actual	1.0	0.5	0.5	0.5	0.3	0.4

Parameters found, arbitrary start, J = f(u) = 0.99016

Table IV - 3

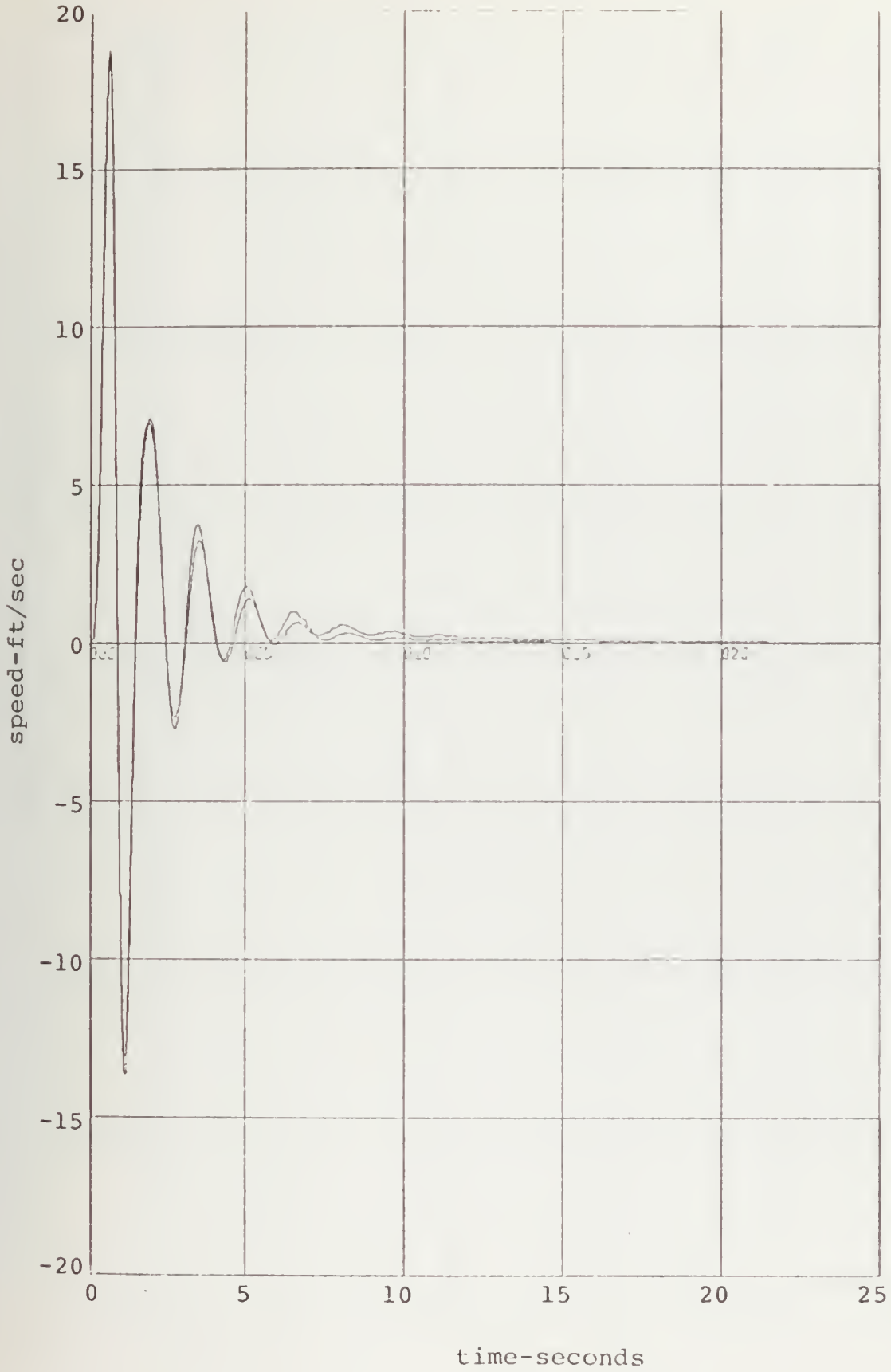


Figure IV - 10

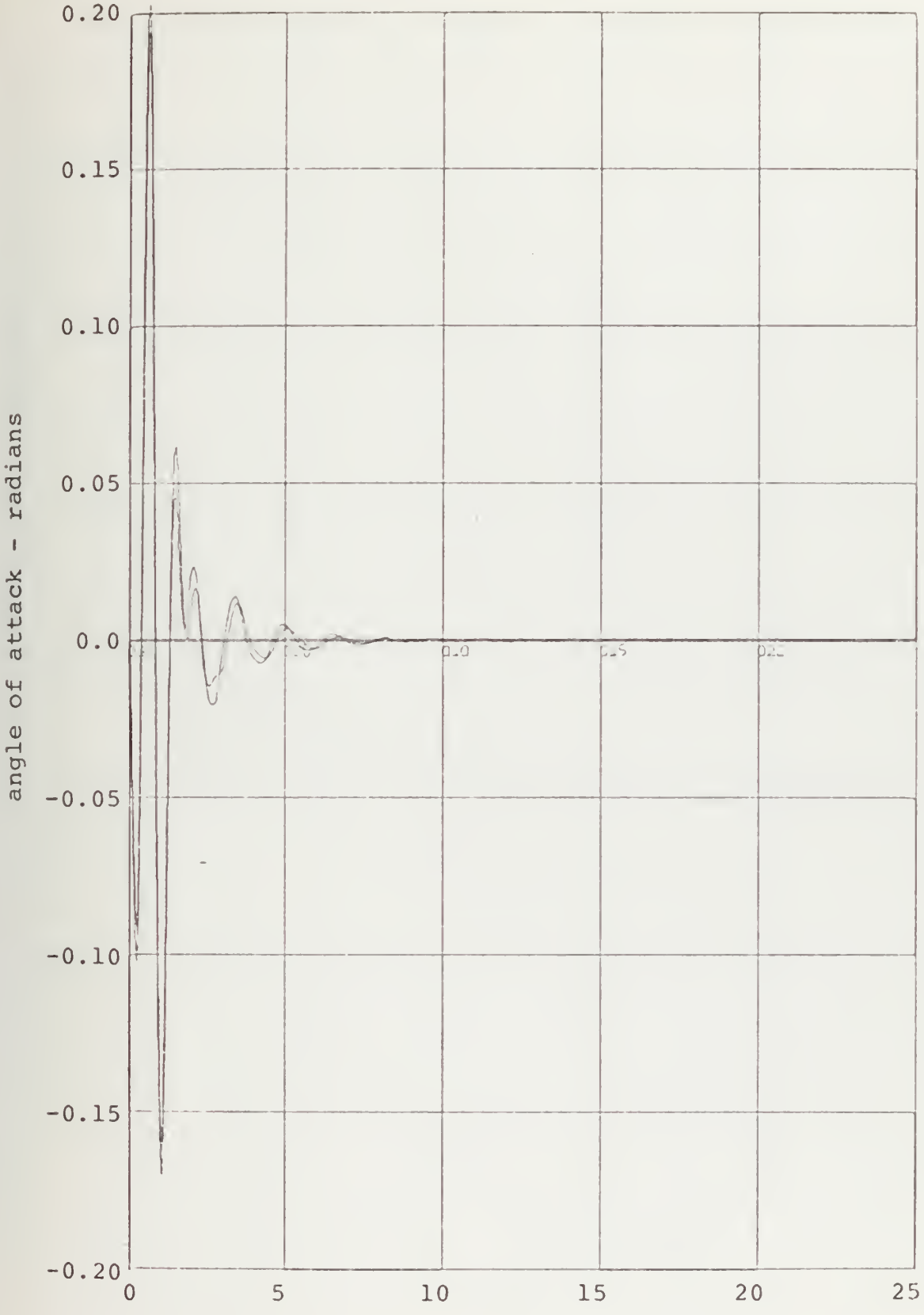

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 11

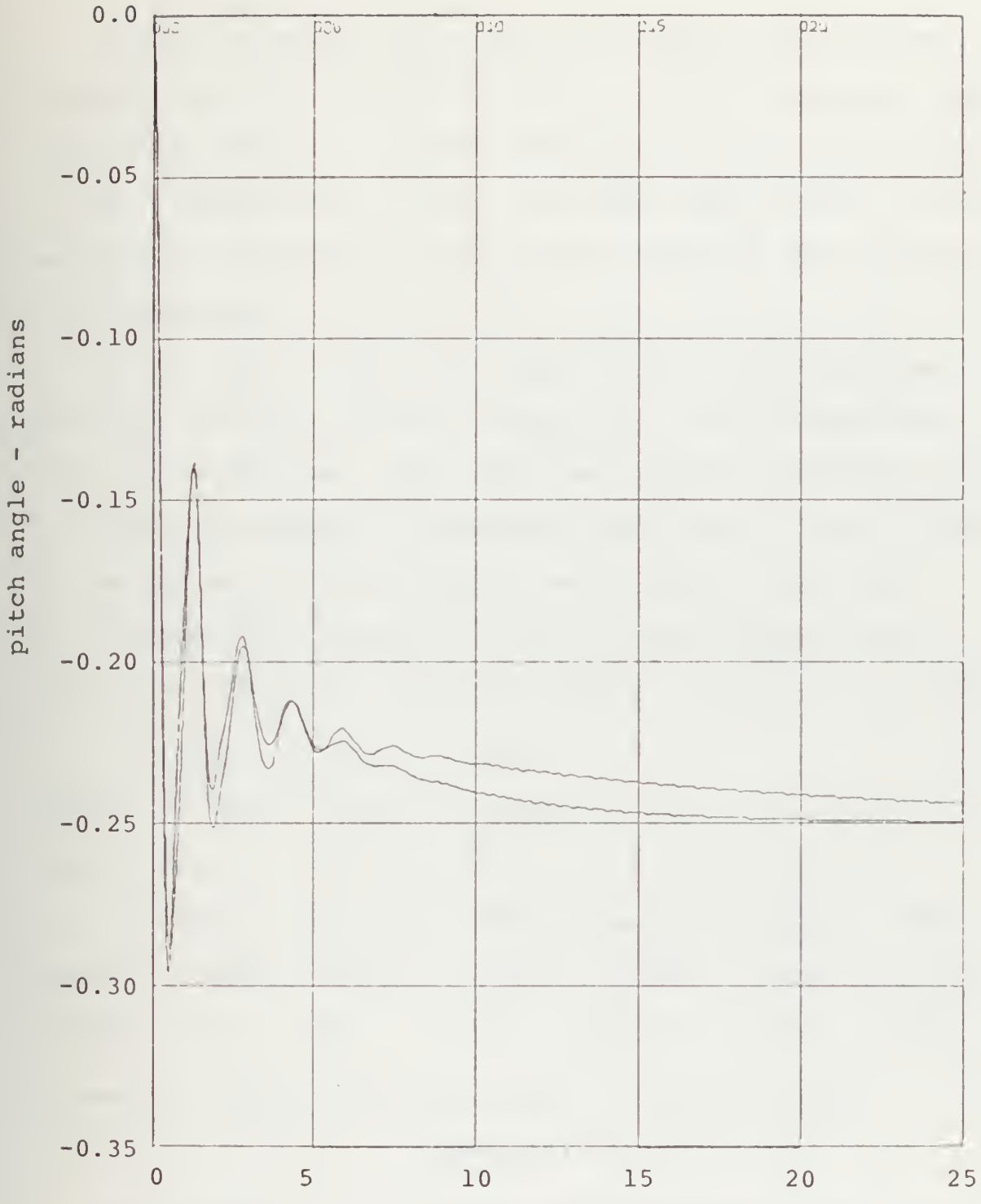

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 12

It will be noted that even though these values differ markedly from the actual ones, the u and α responses are good. But, there is a slowly decreasing steady state error in the response due to the very small value of Zero 5 which has almost cancelled the pole at the origin in the pitch angle compensator.

In the hope of improving these results, the programme was rerun with the values in Table IV - 3 as the starting point, with the same upper and lower bounds (the bounds could be changed to narrow the feasible region, but it was decided to see how well the convergence went without doing this). In a further 555 trials or a total of 1291 BOXPLX came up with a J = .08477 and the values shown in Table IV - 4 below.

	Pole 1	Pole 2	Pole 3			
Found	8.628	5.341	11.150			
Actual	4.0	10.0	10.0			
	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Found	0.3000	1.589	0.497	0.870	0.000	0.912
Actual	1.0	0.5	0.5	0.5	0.3	0.4

Parameters found, arbitrary start, J = f(u) = 0.0848

Table IV - 4

The responses of the system with these poles and zeros are shown in Figures IV - 13, IV - 14, and IV - 15 for u , α and θ respectively.

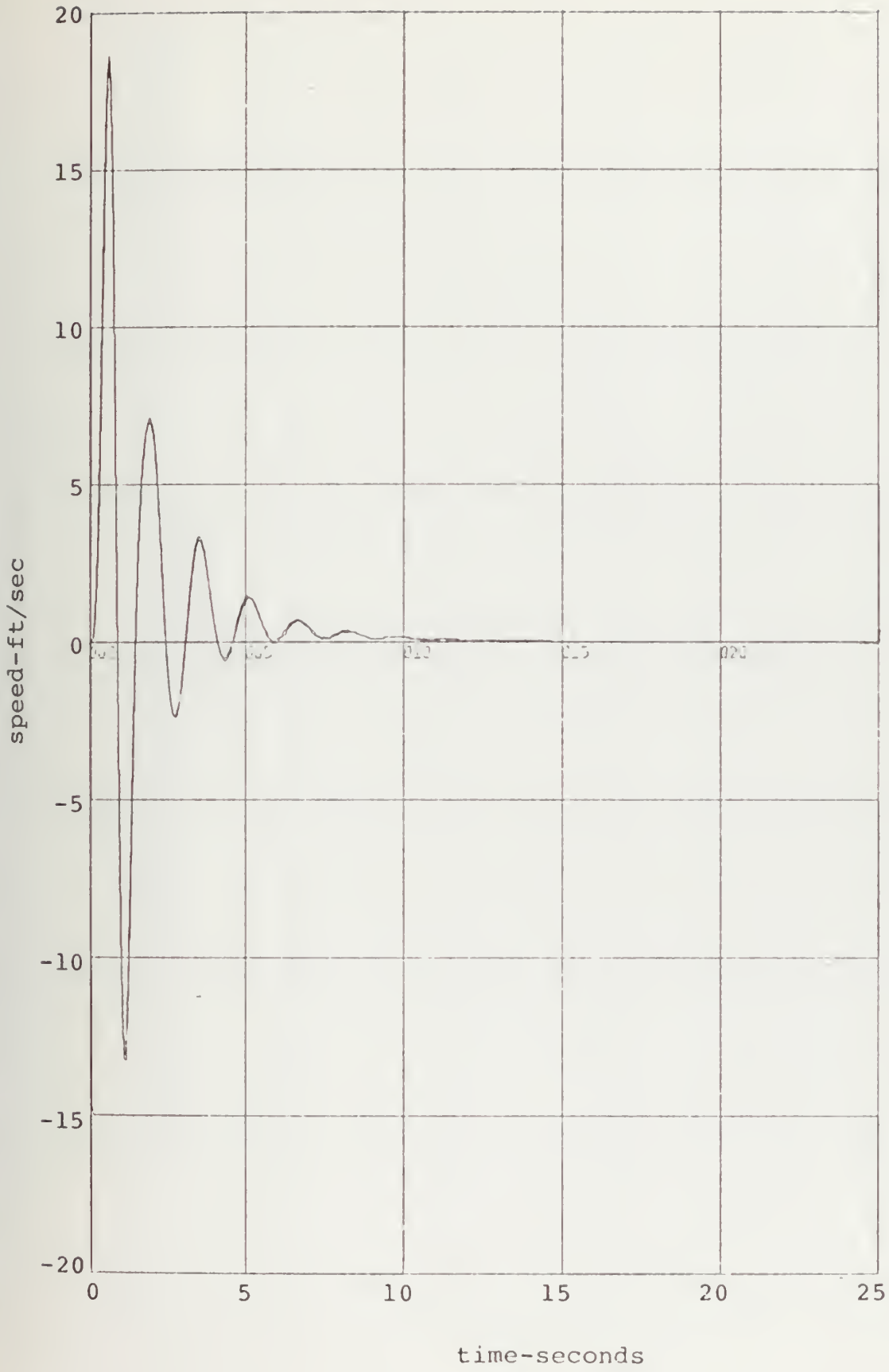


Figure IV - 13

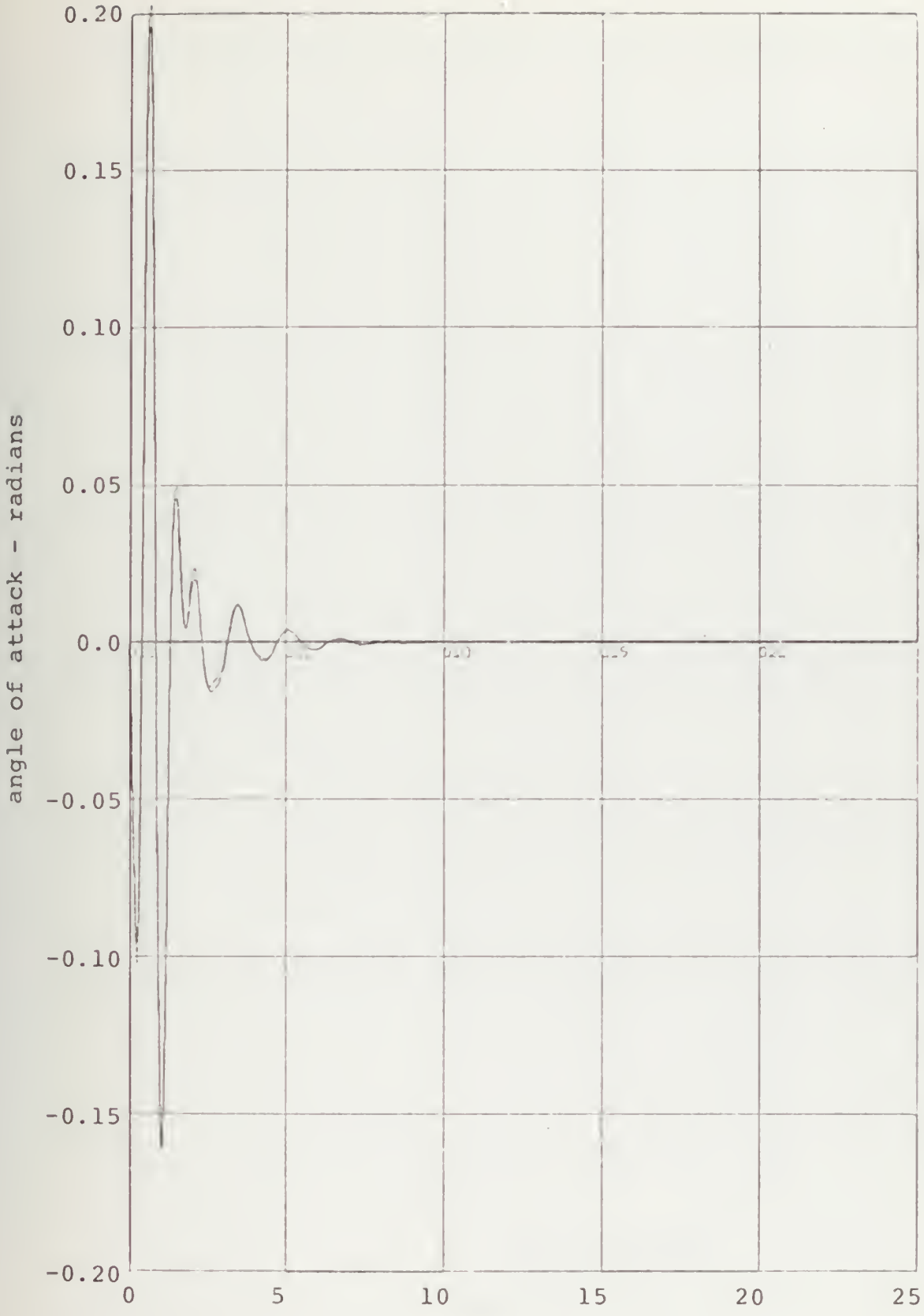

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 14

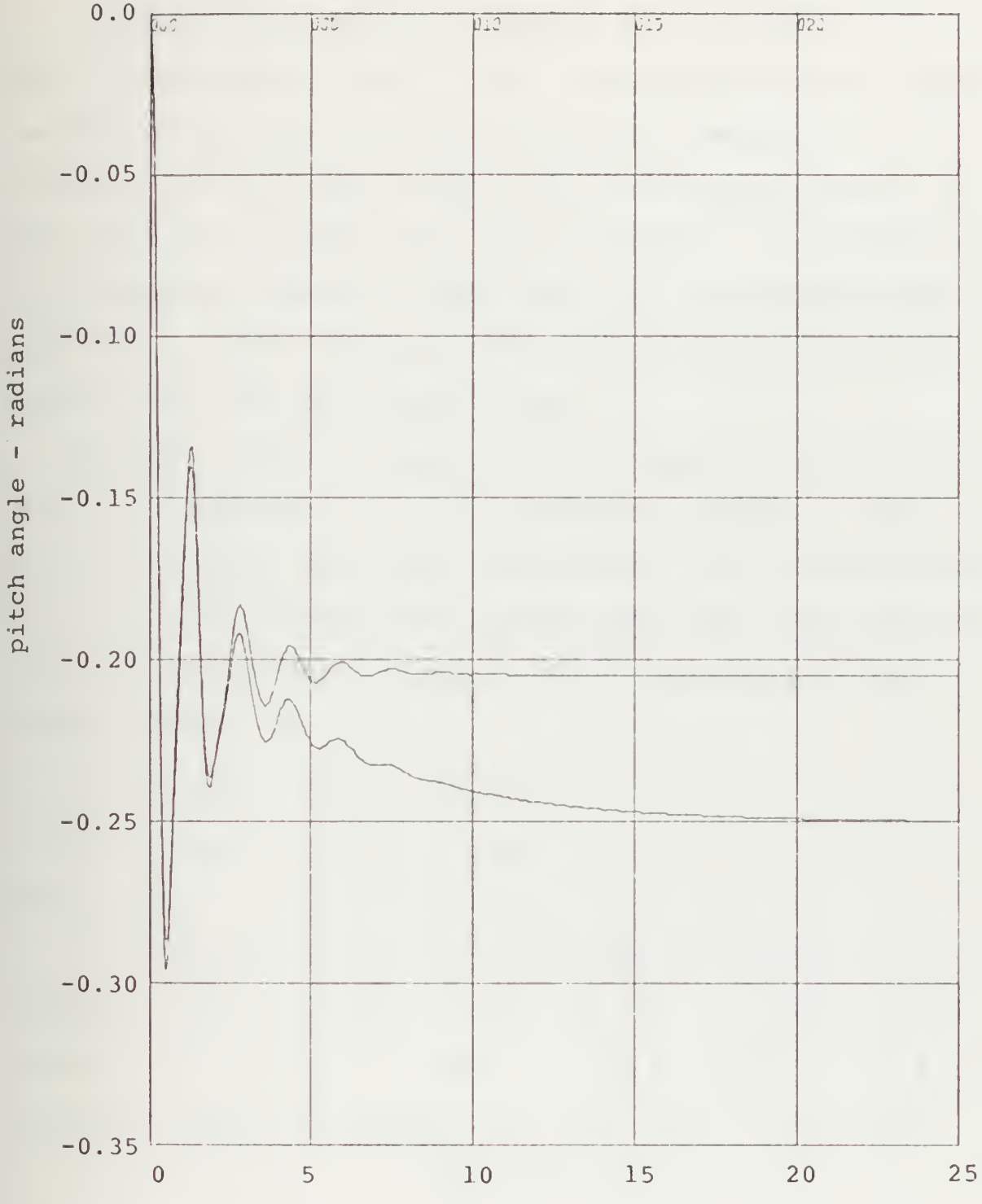

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 15

It is to be noted that, although J is smaller by an order of magnitude and the u and α responses are well nigh a perfect match, there is now a steady state error in the θ response due to a cancellation of the pole at the origin in the pitch angle compensator. This situation is not likely to improve with further trials with J a function of speed. This is the first indication of a worse response with more trials which will arise again later.

The same arbitrary start point was used on the next run with J a function of θ (the commanded output). This time, the results were more encouraging. In 900 trials with $J - 4.1 \times 10^{-6}$ (again not directly comparable with the value when J is a function of speed) BOXPLX came up with the values shown it Table IV - 5.

	Pole 1	Pole 2	Pole 3			
Found	5.259	8.708	9.981			
Actual	4.0	10.0	10.0			
	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Found	0.467	0.249	1.682	0.447	0.325	0.374
Actual	1.0	0.5	0.5	0.5	0.3	0.4

Parameters found, arbitrary start, $J = f( ) = 4.1 \times 10^{-6}$

Table IV - 5

The responses for u , α and θ are plotted in Figures IV - 16, IV - 17 and IV - 18 respectively. The match is near perfect for α and θ and quite acceptable for u . This run reinforced the earlier assertion that better results

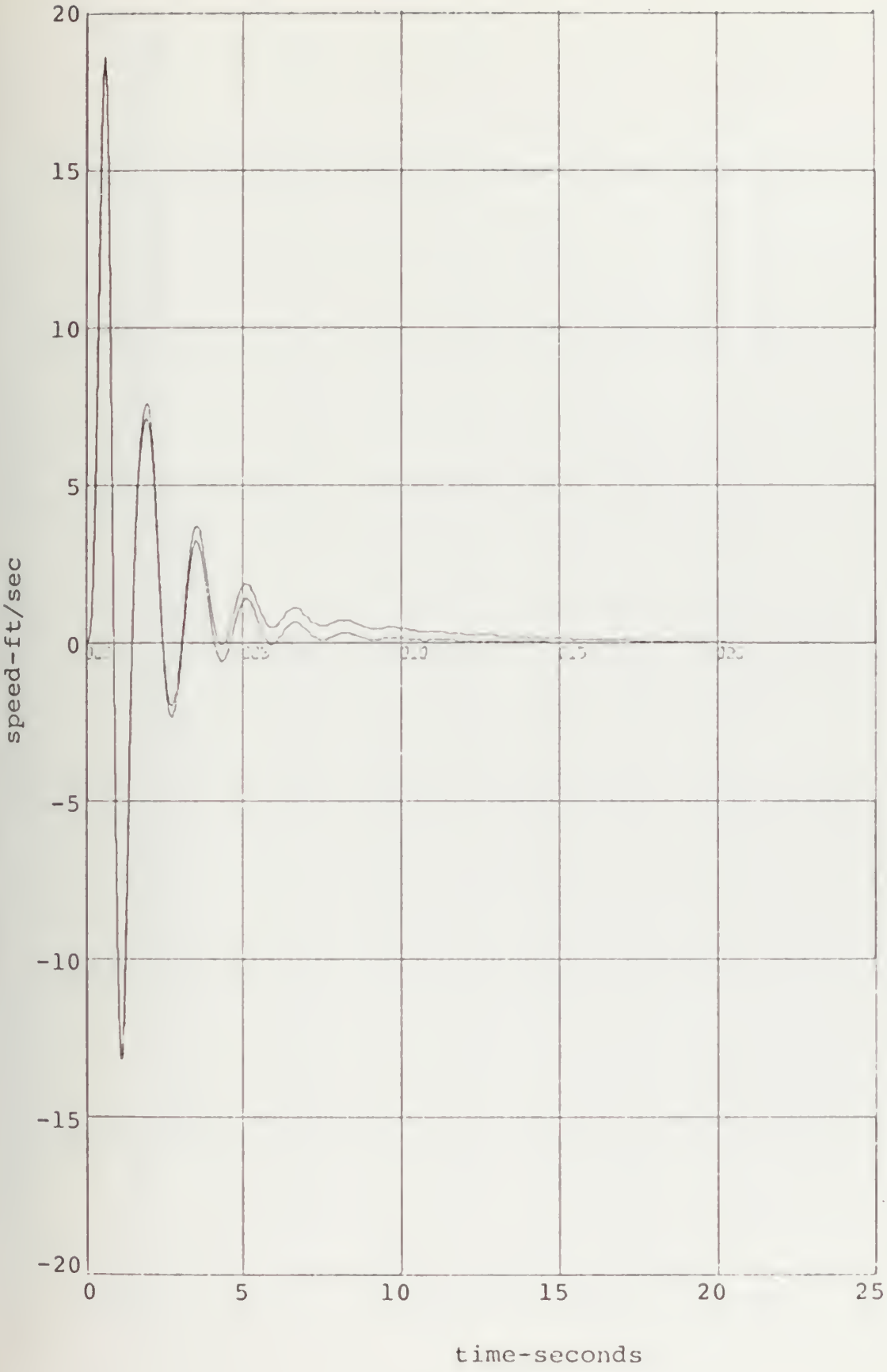


Figure IV - 16

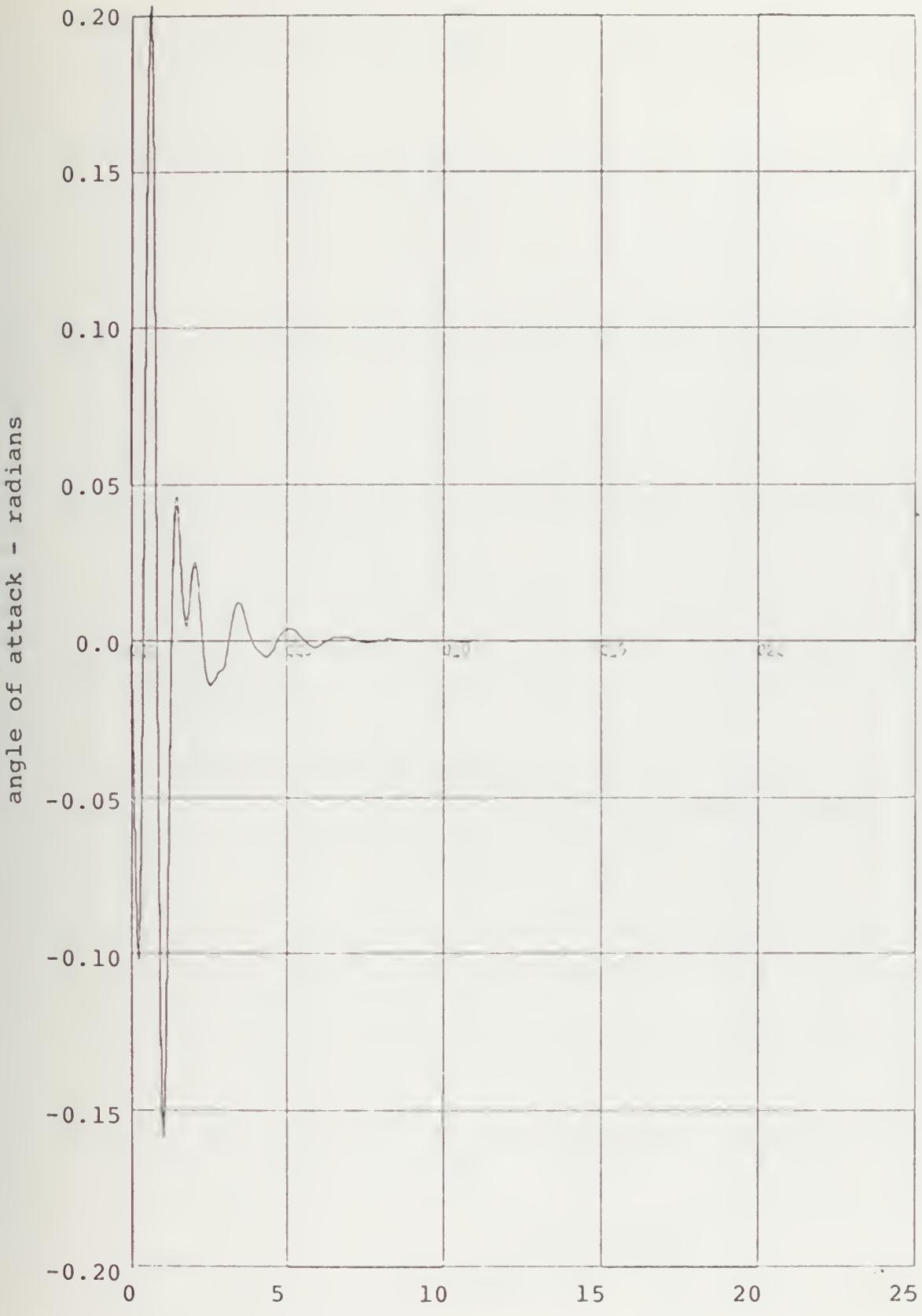

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 17

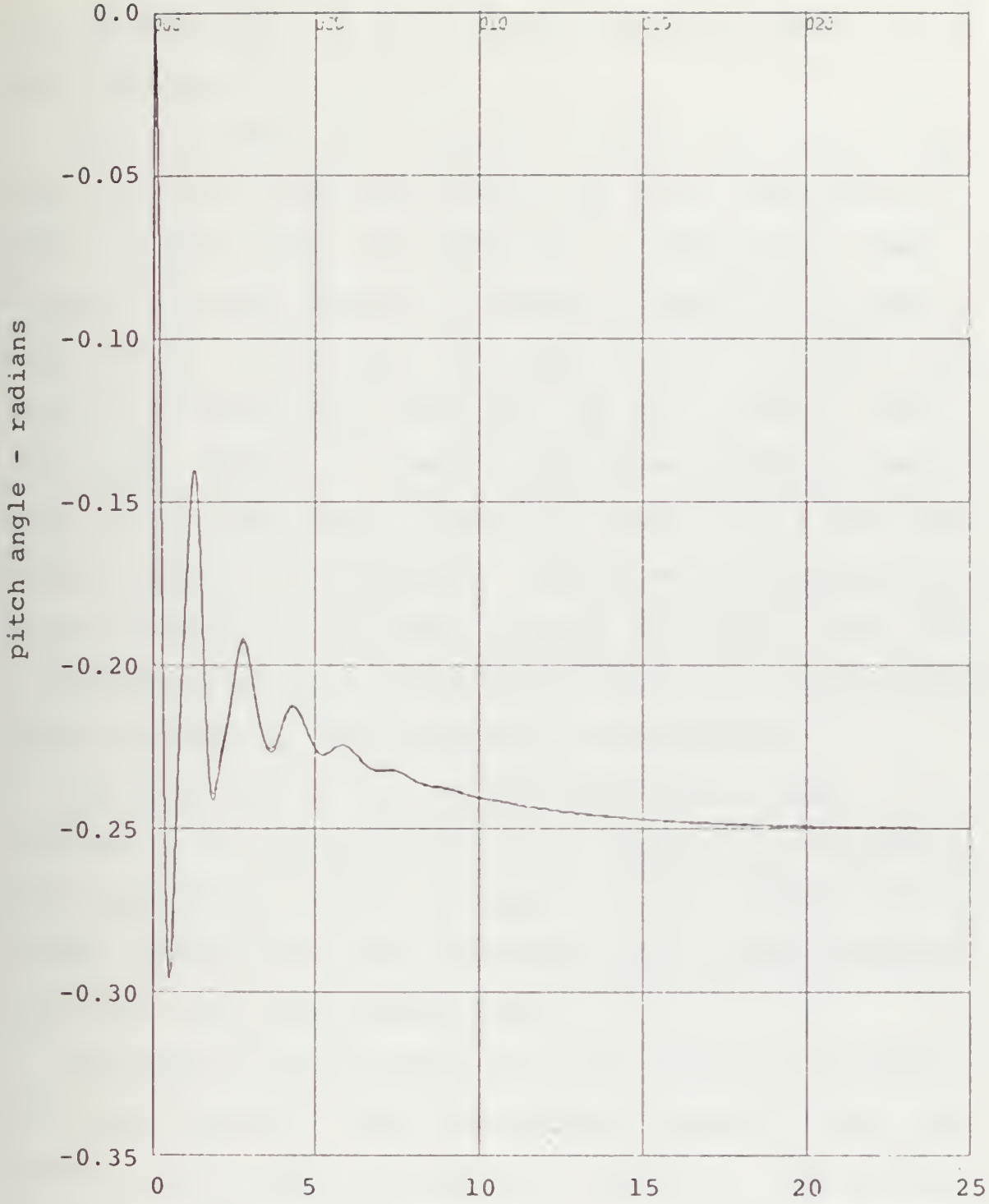

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 18

are obtained when J is a function of the commanded output (at least for this system).

One new wrinkle which appeared during this last run was the case where BOXPLX was not able to find a feasible vertex at a given trial and restarted to compute at the best vertex. However, during the course of computation after this restart, it was again not able to find a feasible vertex and the best vertex was still the one at which it had restarted. Thus exit from the subroutine occured. The remedy in this case was to change the value of R , the first random number of the sequence generated in developing the initial complex of vertices. It must be admitted that this is probably a matter of luck as it is with all random numbers, but the results of the run speak for themselves.

The last test in this initial phase was to make J a function of all three outputs as an example of this admittedly artificial situation, since it is not likely that, for a given problem, one would know what all the outputs should look like, much less specify them.

The problem arising here is to put weighting factors in the cost function so that the response variations from the desired response may be penalized depending on the importance of one or more outputs over others. If all are considered equally important which would normally be the case if they were specified it remains only to balance the cost function component variations due to scale factors. This was fairly

easy in this case because of the data from past runs. The value of J for the arbitrary start point when J was a function of speed was 0.9989046×10^{51} and the equivalent for pitch angle was 0.7685555×10^{47}. A trial run with J a function of all three outputs without weighting factors produced $J = 0.9990991 \times 10^{51}$. Subtracting the first two from the last resulted in a J for alpha = 0.1176444×10^{48}. Thus as a rough approximation, the speed component of the cost function was made 0.001, the angle of attack component, 0.1 and the pitch angle component, 1.0.

With this cost function, the best vertex after 1120 trials with J = .00042 was as shown in Table IV - 6.

	Pole 1	Pole 2	Pole 3			
Found	6.339	7.203	10.597			
Actual	4.0	10.0	10.0			
	Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
Found	1.003	0.252	1.098	0.641	0.240	0.524
Actual	1.0	0.5	0.5	0.5	0.3	0.4

Parameters found, arbitrary start, $J = f(u,\alpha,\theta) = 0.00042$

Table IV - 6

The system responses for these poles and zeros are plotted in Figures IV - 19, IV - 20 and IV - 21 for u, α and θ respectively. The response matching is almost perfect although the pole and zero values differ considerably from the actual values.

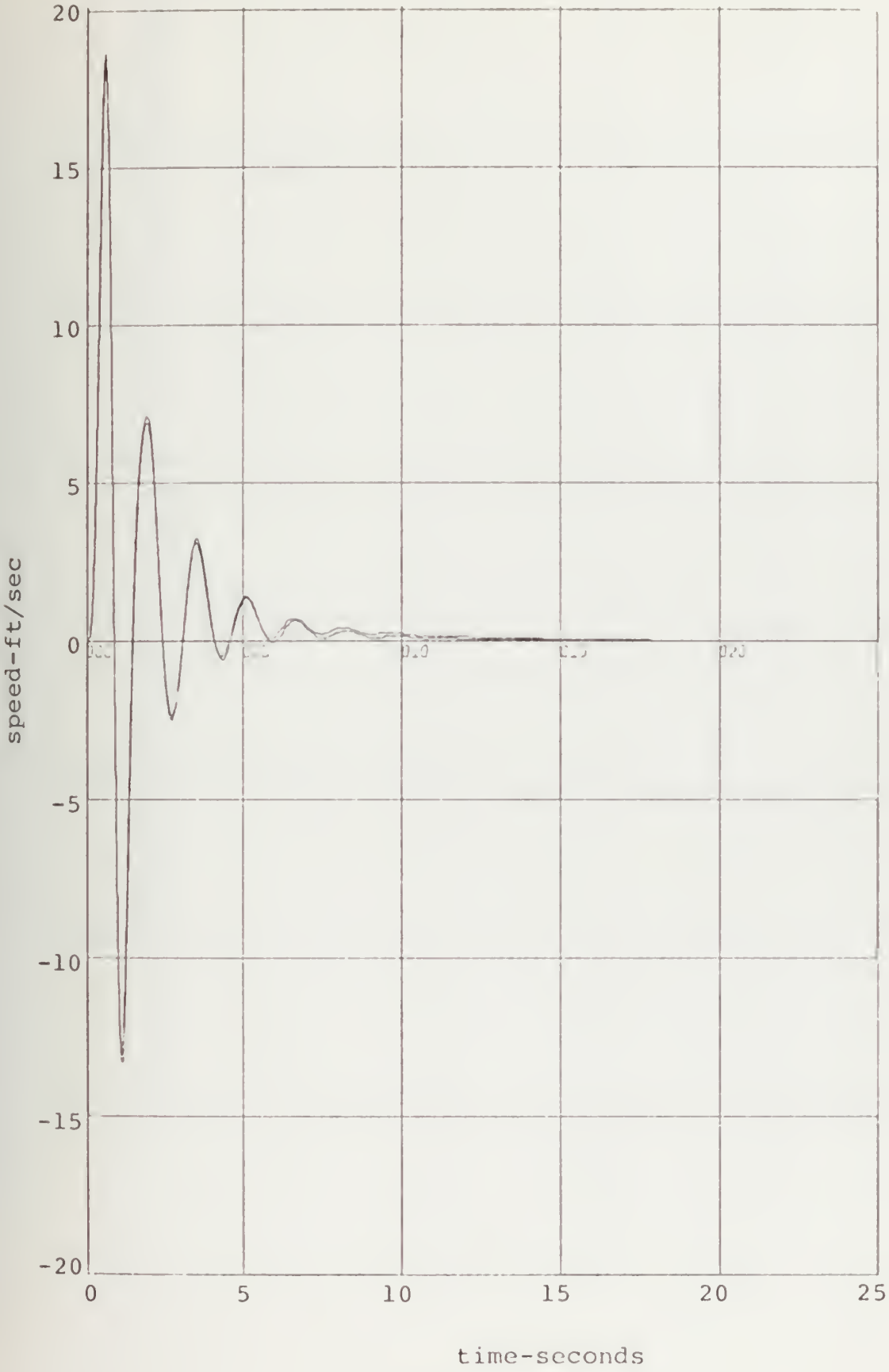


Figure IV - 19

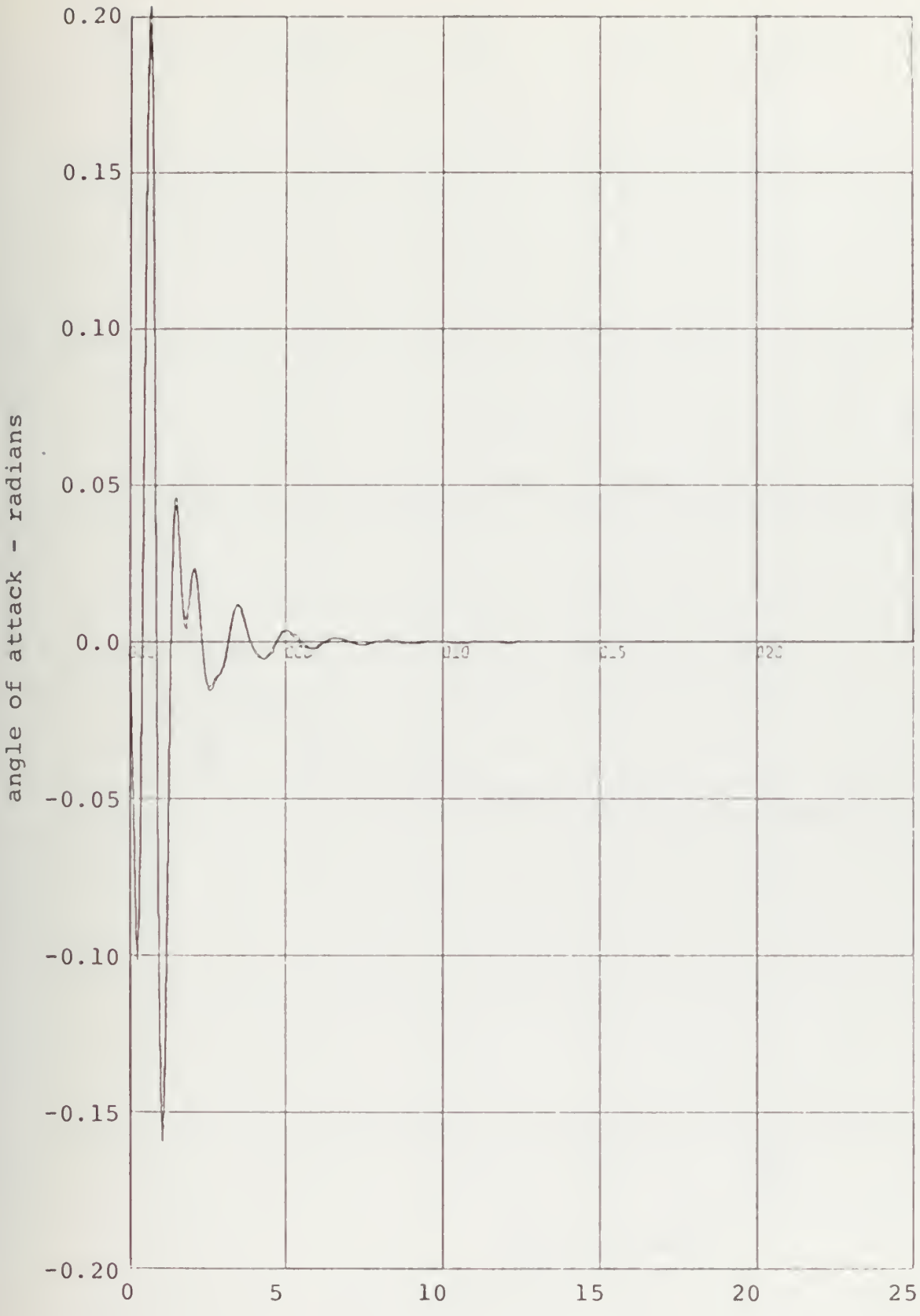

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 20

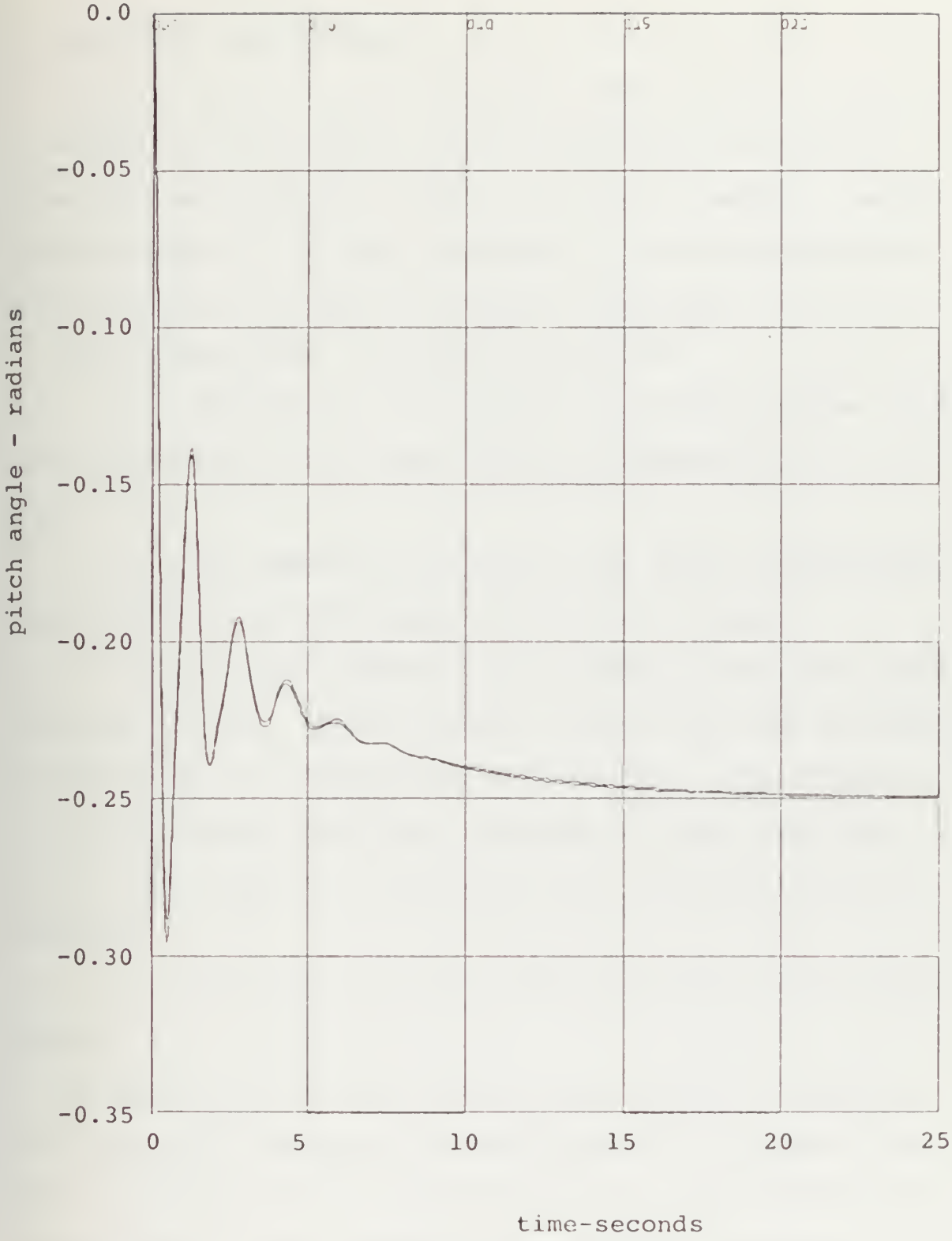

0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
-0.35
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 21

B. ARBITRARY DESIGN PHASE

At this point, it was considered that the intricacies of BOXPLX and the factors affecting desired results were understood well enough to form a strategy by which a multivariable system of a given configuration could be designed to have any time response desired to any input.

The strategy was formulated as follows:

a. Specify the time response and make the cost function a function of the output which is commanded by the input.

b. Let BOXPLX try to match that output response for 600 to 900 trials and examine plots of all outputs.

c. If other outputs are reasonable, success has been achieved. If not, specify another output which may be important and make J a function of both outputs. This other output specification should be a compromise between what the system seems capable of achieving and a reasonable desired response.

d. Continue iteration until all responses are satisfactory.

Consequently, for this system it was decided to specify the pitch angle response to a step input of -0.25 for r_3 as shown in Figure IV - 22. This particular step response was obtained from a state variable simulation of Whiteley's standard form for a step response with a maximum of 10% overshoot on zero position error (Ref. 7) with an $\omega_o = 8.75$.

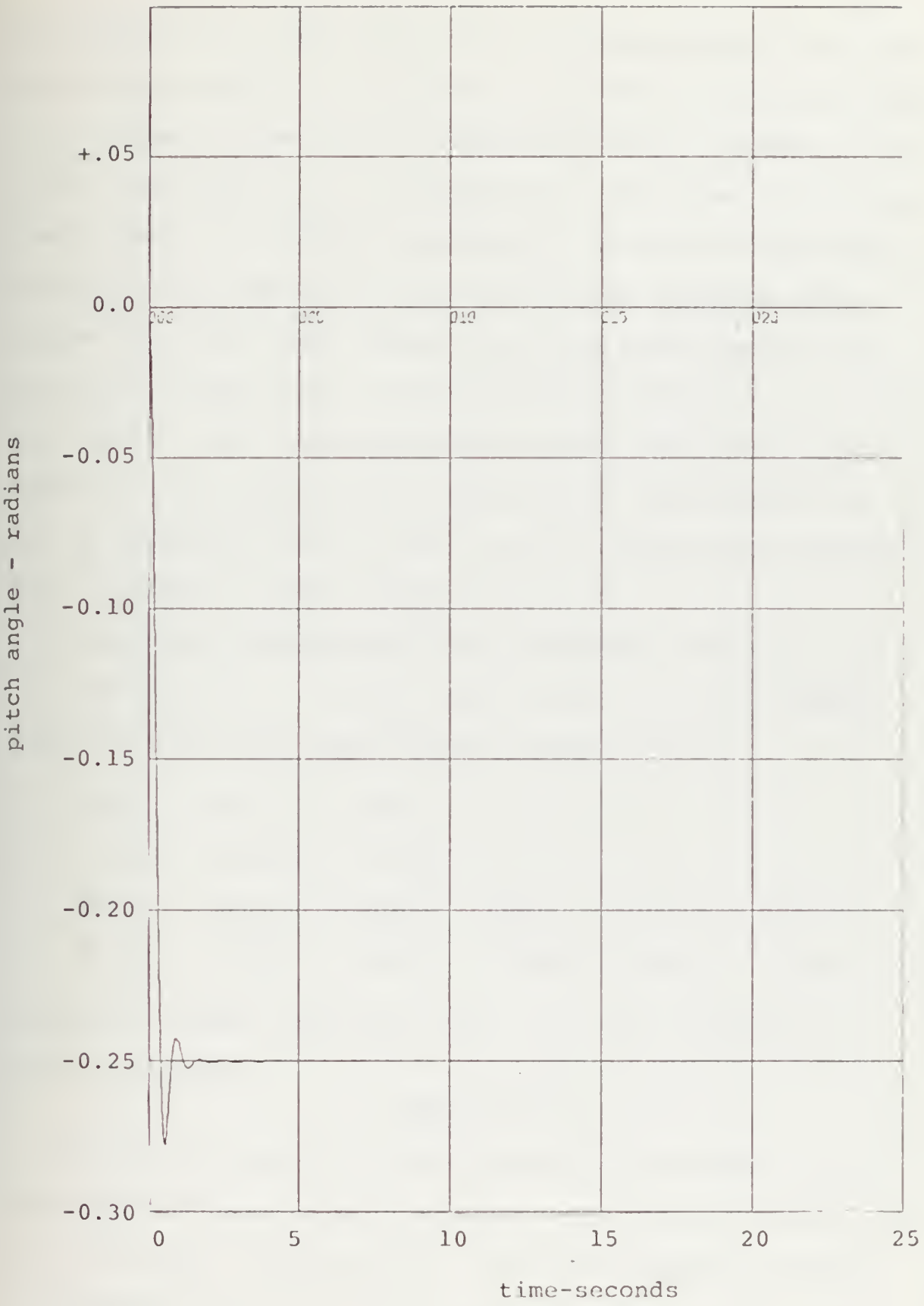

+.05
0.0
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 22

The value of ω_o was chosen from an examination of the natural time constant of the system as revealed by previous data, which showed a rise time of approximately 0.5 seconds. Initials runs with $\omega_o = 2.5$ which gave a rise time of 1.75 seconds showed that BOXPLX was having a difficult time trying to design the compensator for such a slow response. The values for Pole 3 kept ending up on the upper bounds, and they were raised twice to 30.0 and 40.0 before an $\omega_o = 8.75$ was chosen. The computer program at the end of this paper shows how the response was simulated and cards punched to act as reference input to the function minimization program. This response is shown in Figure IV - 22.

Thus a run was made with this response a reference and J function of θ . In 900 trials with J = .00156, BOXPLX found the poles and zeros shown in Table IV - 7.

Pole 1	Pole 2	Pole 3			
3.669	14.644	19.890			
Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
0.802	0.397	0.034	2.054	0.752	1.220

Parameters found, arbitrary start, J = f(θ) = .00156, arbitrary response

Table IV - 7

The response curves of the system with the above poles and zeros are plotted in Figures IV - 23, IV - 24 and IV - 25 for u , α and θ respectively. The theta response matches the reference arbitrary response fairly well and the alpha

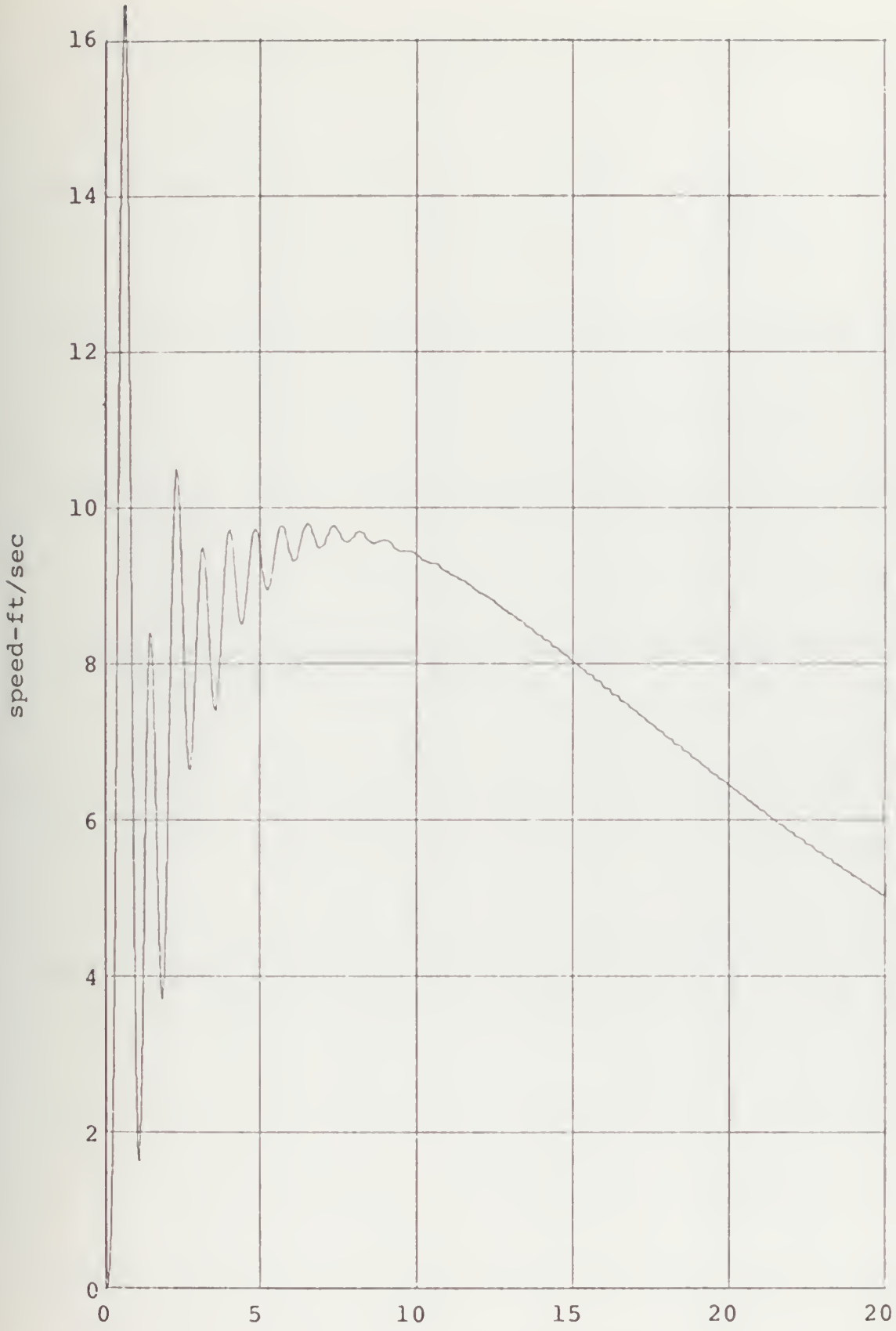


Figure IV - 23

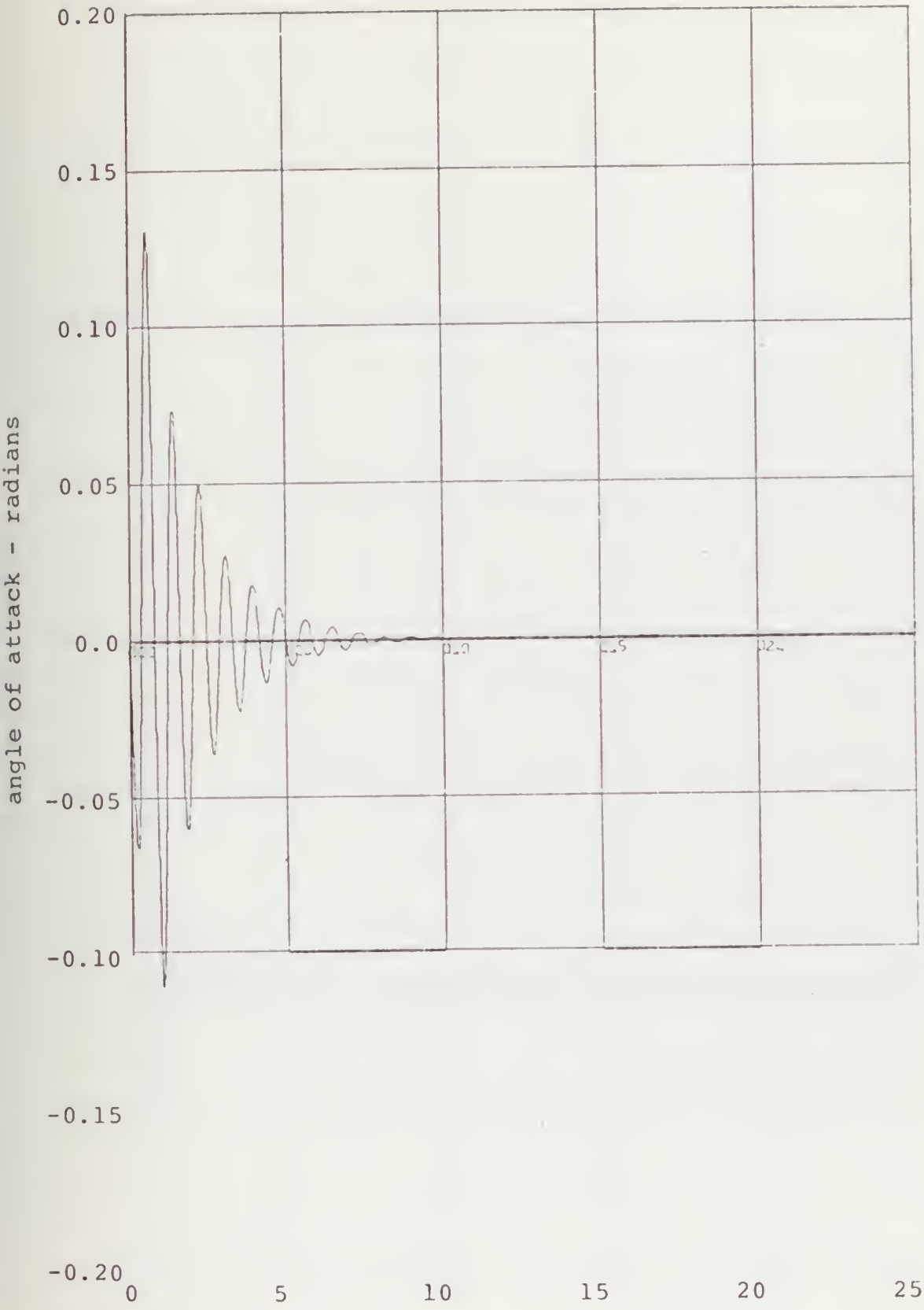

0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.20
0
5
10
15
20
25
angle of attack - radians
time-seconds

Figure IV - 24

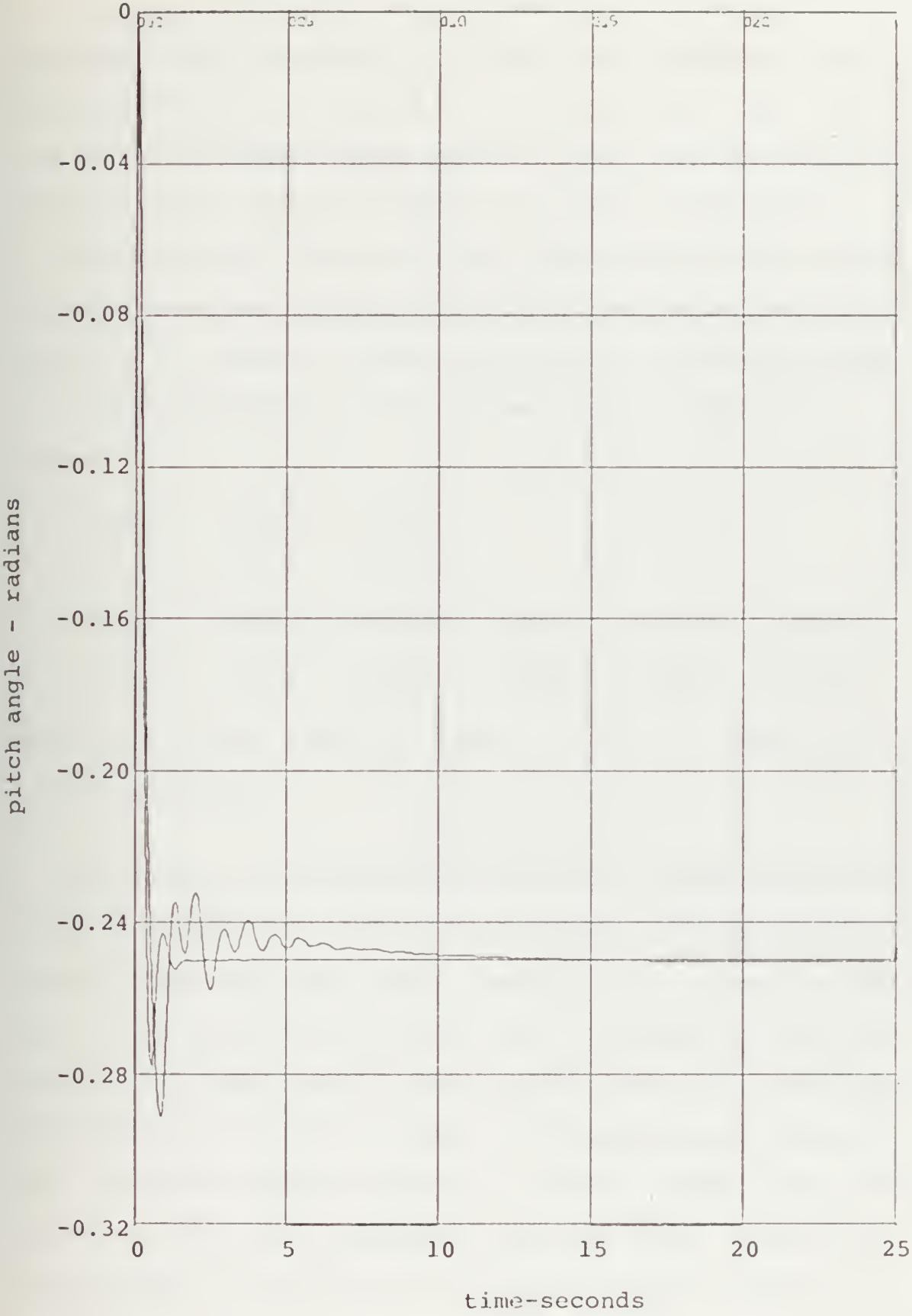

0
-0.04
-0.08
-0.12
-0.16
-0.20
-0.24
-0.28
-0.32
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 25

response looks reasonable but there is an extremely long settling time for the speed to return to zero. This was due to the extremely small value of Zero 3 in the speed compensator which almost cancelled its pole at the origin.

In the hopes that more trials might improve the results the programme was run again starting from the values in Table IV - 7. After a further 900 trials or 1800 total, with J = .00088, BOXPLEX found the values give in Table IV - 8 below:

Pole 1	Pole 2	Pole 3			
7.173	8.635	14.933			
Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
0.884	0.243	0.000	0.000	0.809	0.810

Parameters found, arbitrary start, J = f(0) = .00088, arbitrary response.

Table IV - 8

The results of a system simulation with these values are shown in Figures IV - 26, IV - 27 and IV - 28 for speed, angle of attack and pitch angle respectively. It can be noted that the match to the arbitrary theta response is excellent and that the short period oscillations both in it and in the alpha response have been reduced in amplitude and duration. But, the speed response now has a constant steady state error due to the fact that decoupling has been lost. It will be recalled that decoupling of this multi-variable system required a pole at the origin in each of the compensators, but,

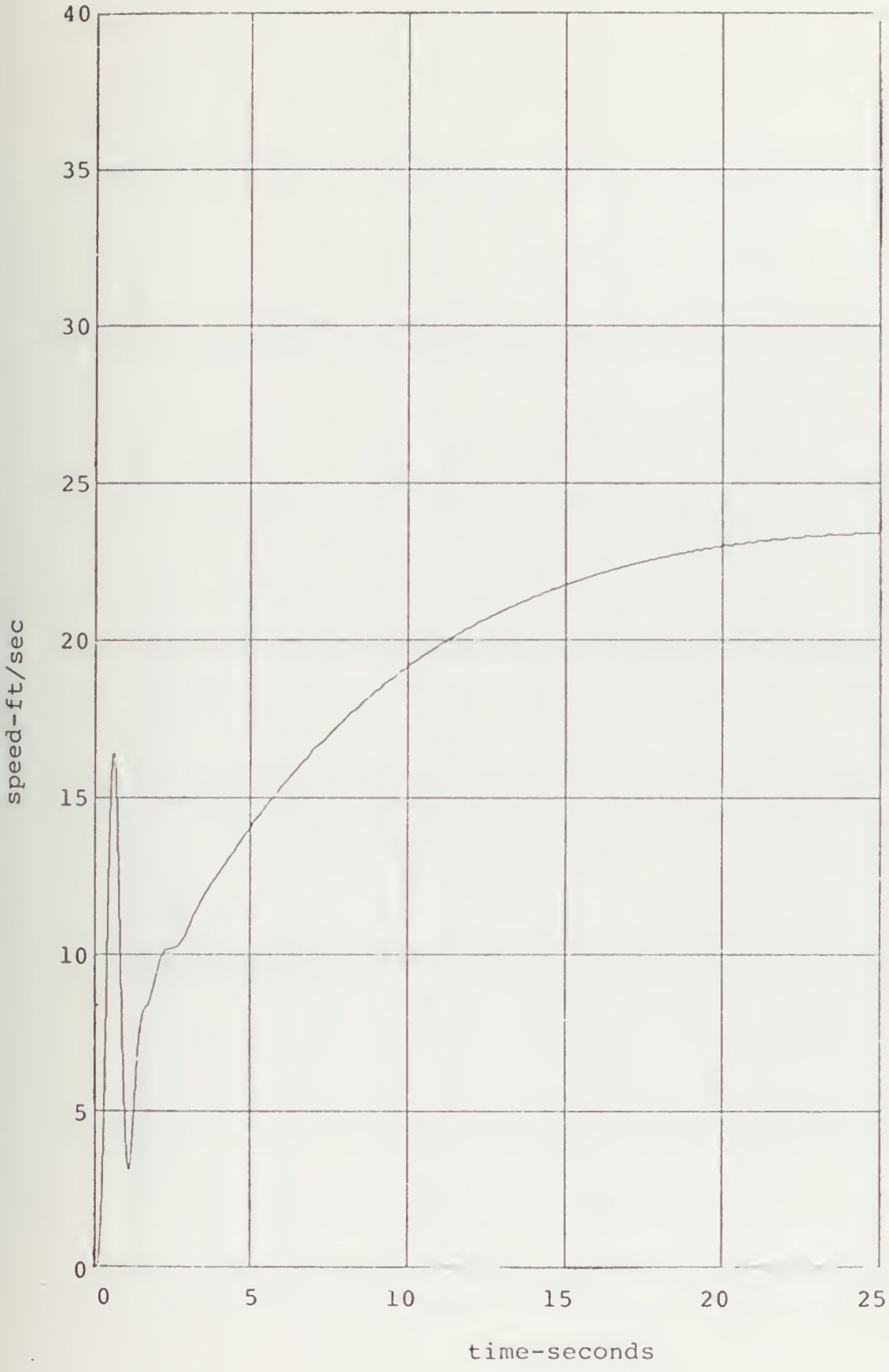


Figure IV - 26

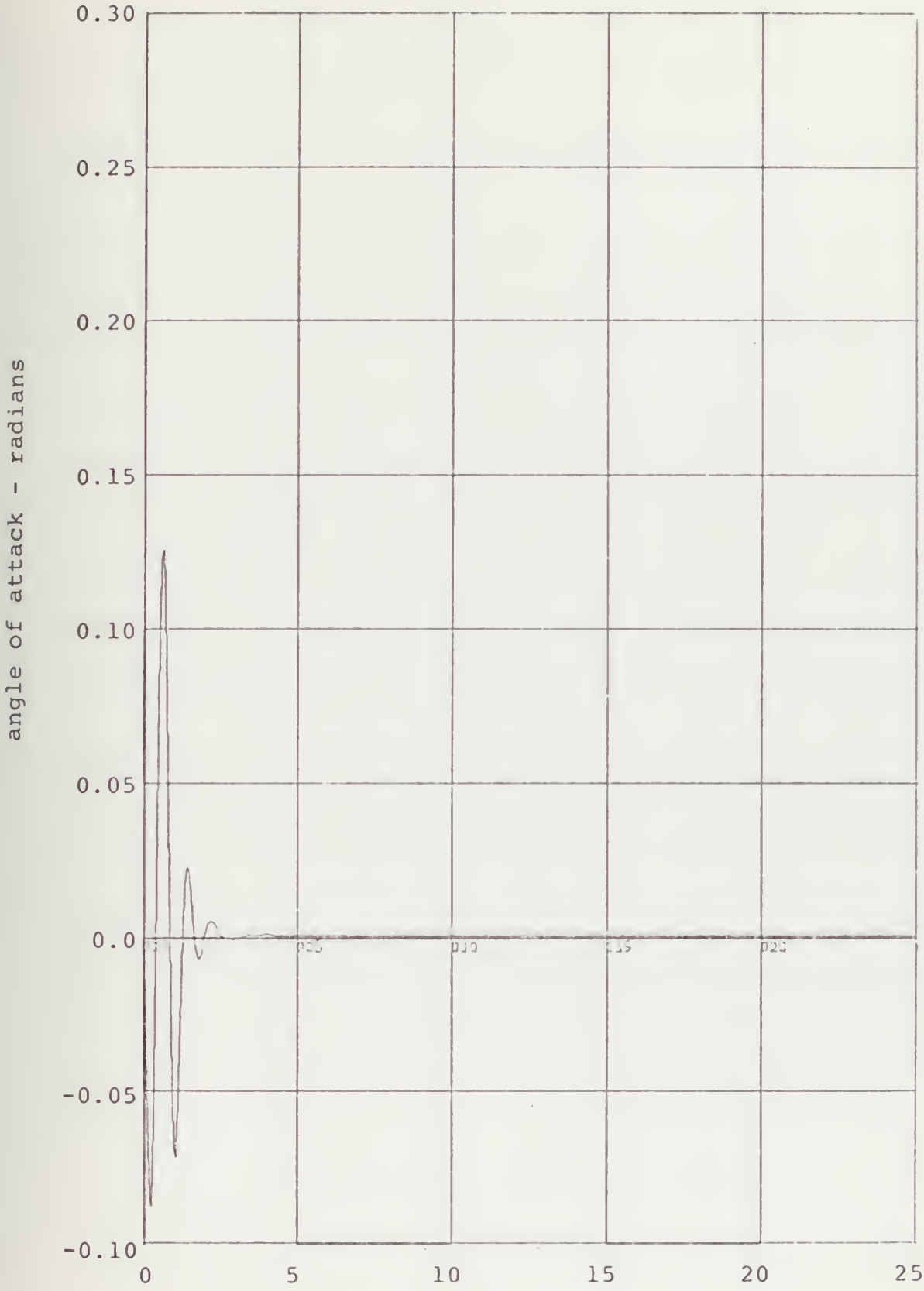

0.30
0.25
0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 27

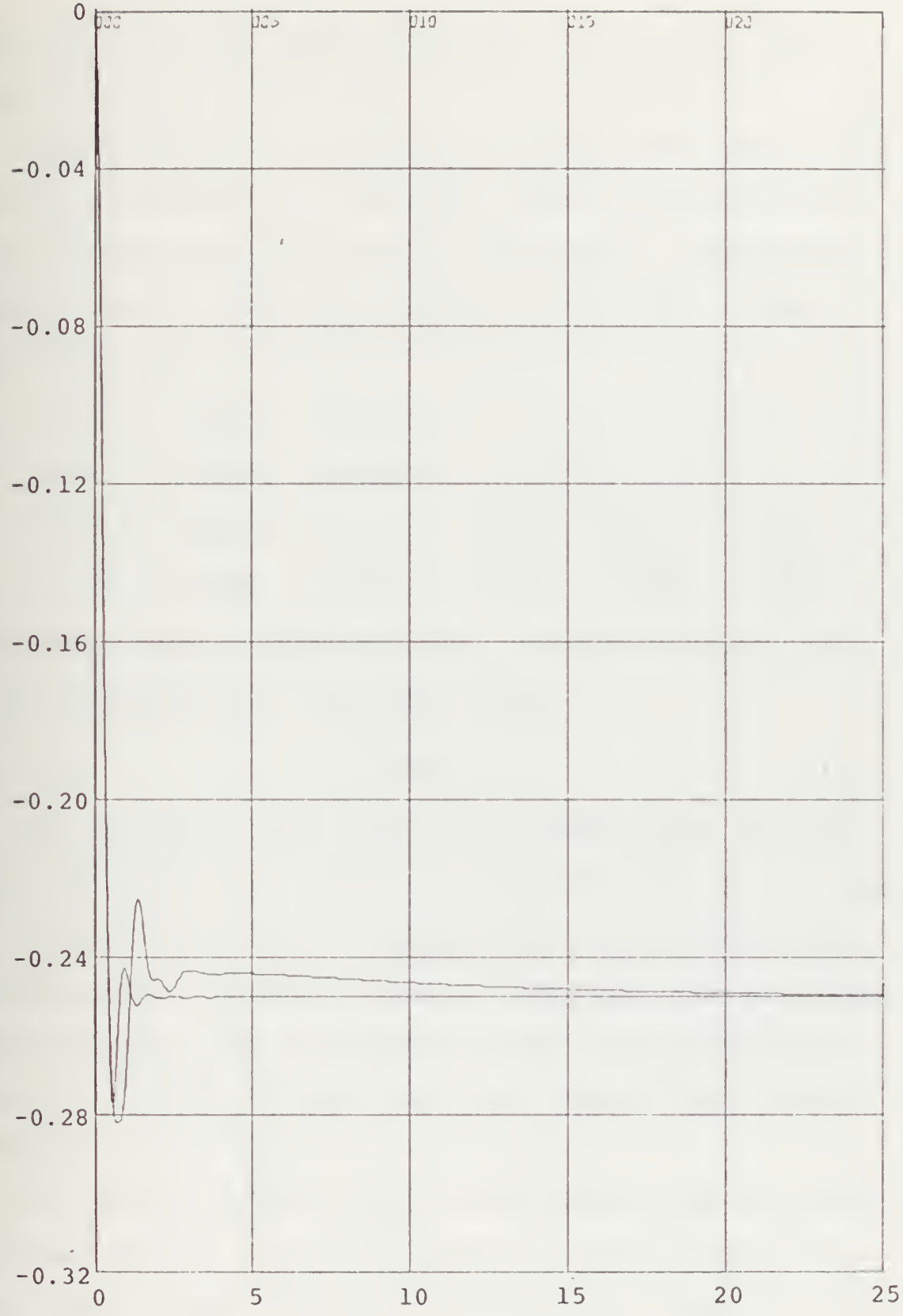

0
-0.04
-0.08
-0.12
-0.16
-0.20
-0.24
-0.28
-0.32
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 28

since Zero 3 in the speed compensator went to the lower bound of 0.0, it effectively cancelled out this pole.

Consequently, it was decided to change the lower bounds for all the zeros to an arbitrary value of 0.25 and rerun the programme from the arbitary start point. This time, BOXPLX found the following values in Table IV - 9 after 900 trials with J = .00180.

Pole 1	Pole 2	Pole 3			
8.994	19.519	14.510			
Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
0.331	0.298	2.332	0.763	0.942	0.702

Parameters found, arbitrary start, J = $f(\theta)$ = .0018, arbitrary response, BL = .25 on all zeros.

Table IV - 9

The response of the system with these values for the poles and zeros are plotted in Figures IV - 29, IV - 30, and IV - 31 for u , α and θ respectively. The results show some secondary overshoot on the θ response, some increased oscillations in the α response but the speed response is beginning to look like what one would expect from the conditions.

To check if further trials would improve the results the programme was rerun for a further 900 trials starting from the values in Table IV - 9. The results of these 1800 trials with J = .00099 are shown in Table IV - 10 below.

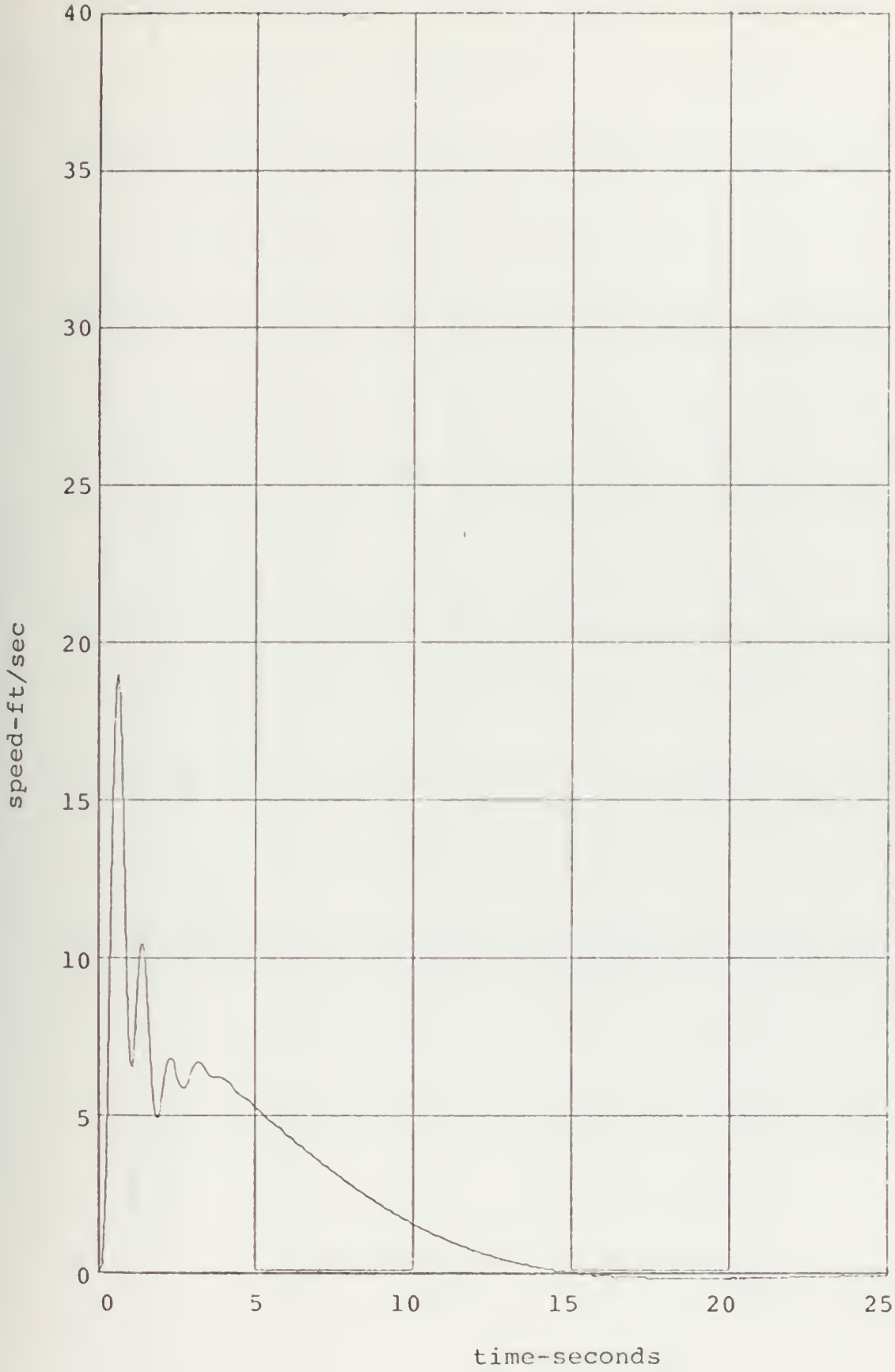


Figure IV - 29

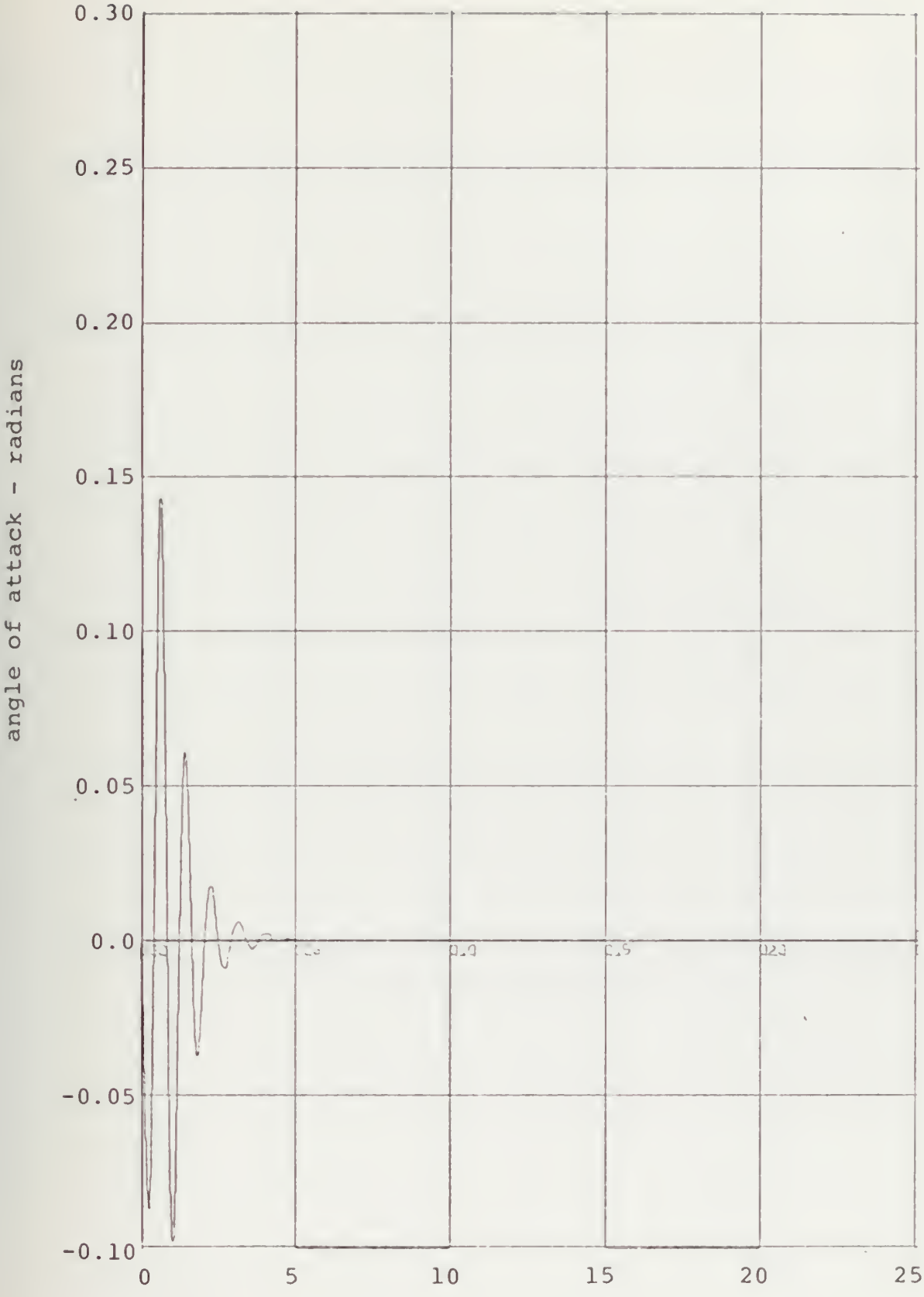

0.30
0.25
0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 30

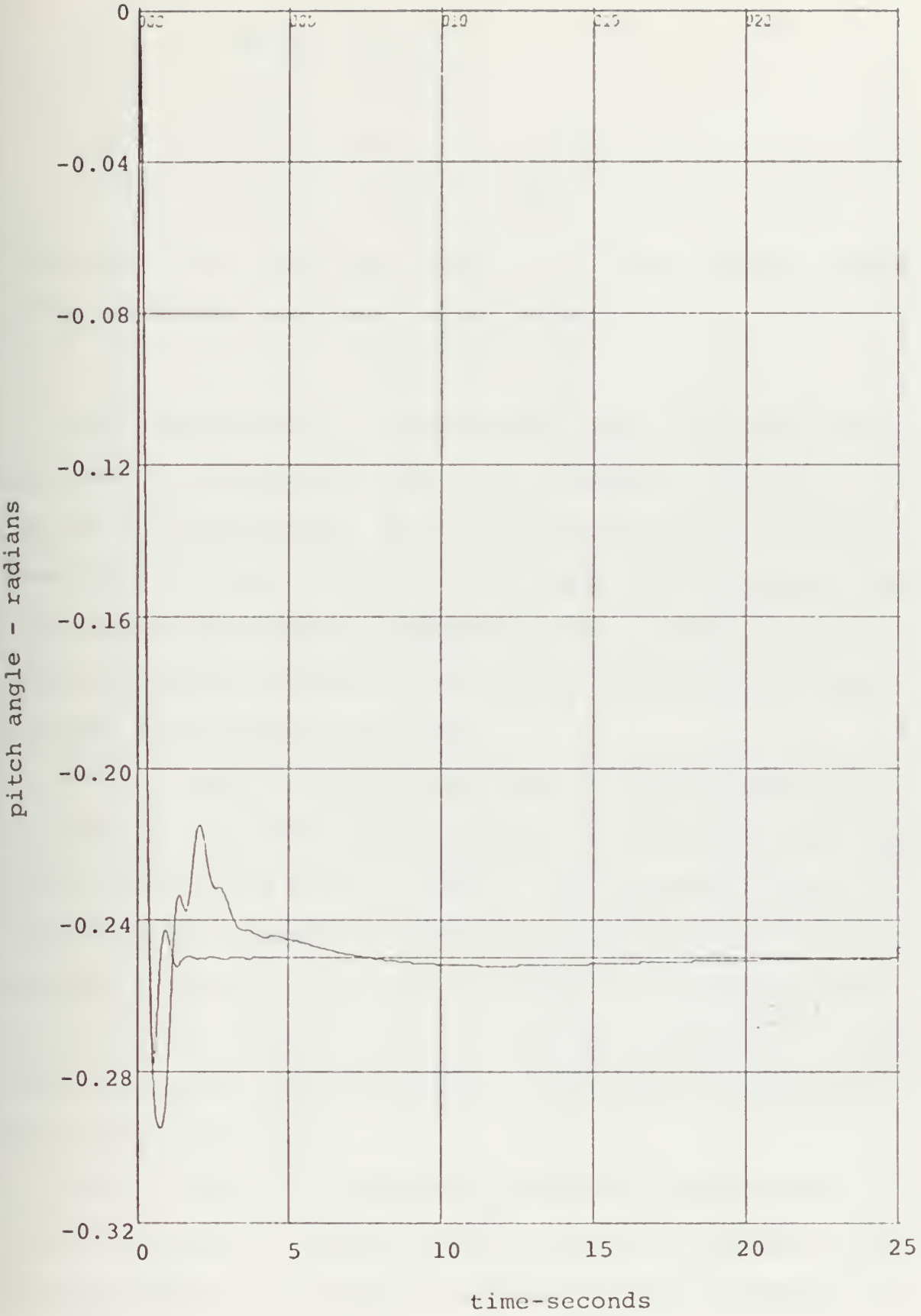

0
-0.04
-0.08
-0.12
-0.16
-0.20
-0.24
-0.28
-0.32
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 31

Pole 1	Pole 2	Pole 3			
7.007	8.231	15.372			
Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
0.240	0.253	0.567	0.240	0.842	0.828

Parameters found arbitrary start, $J = f(\theta) = .00099$, arbitrary response. BL = 0.25 on all zeros.

Table IV - 10

The responses of the system with these poles and zeros are shown in Figures IV - 32, IV - 33 and IV - 34 for u, α and θ respectively. As was mentioned before as being sometimes the case, these further trials did not improve the theta and alpha responses significantly but definitely worsened the speed response to one with a long settling time and some long period oscillation.

Thus it was now time to backtrack to the previous run and specify a suitable speed response and make the cost function a function of both u and θ. The response chosen in concert with colleagues cognizant in this area was the one shown in Figure IV - 35. This response was obtained from a state variable simulation of the impulse response of a second order overdamped system as shown in the computer programme at the end of this paper.

After revising the function minimization programme to the final form shown at the end of this paper as a sample of all such programmes, to include a balanced J as a function of u and θ, BOXPLX, in 1313 trials found the pole and zero values given in Table IV - 11.

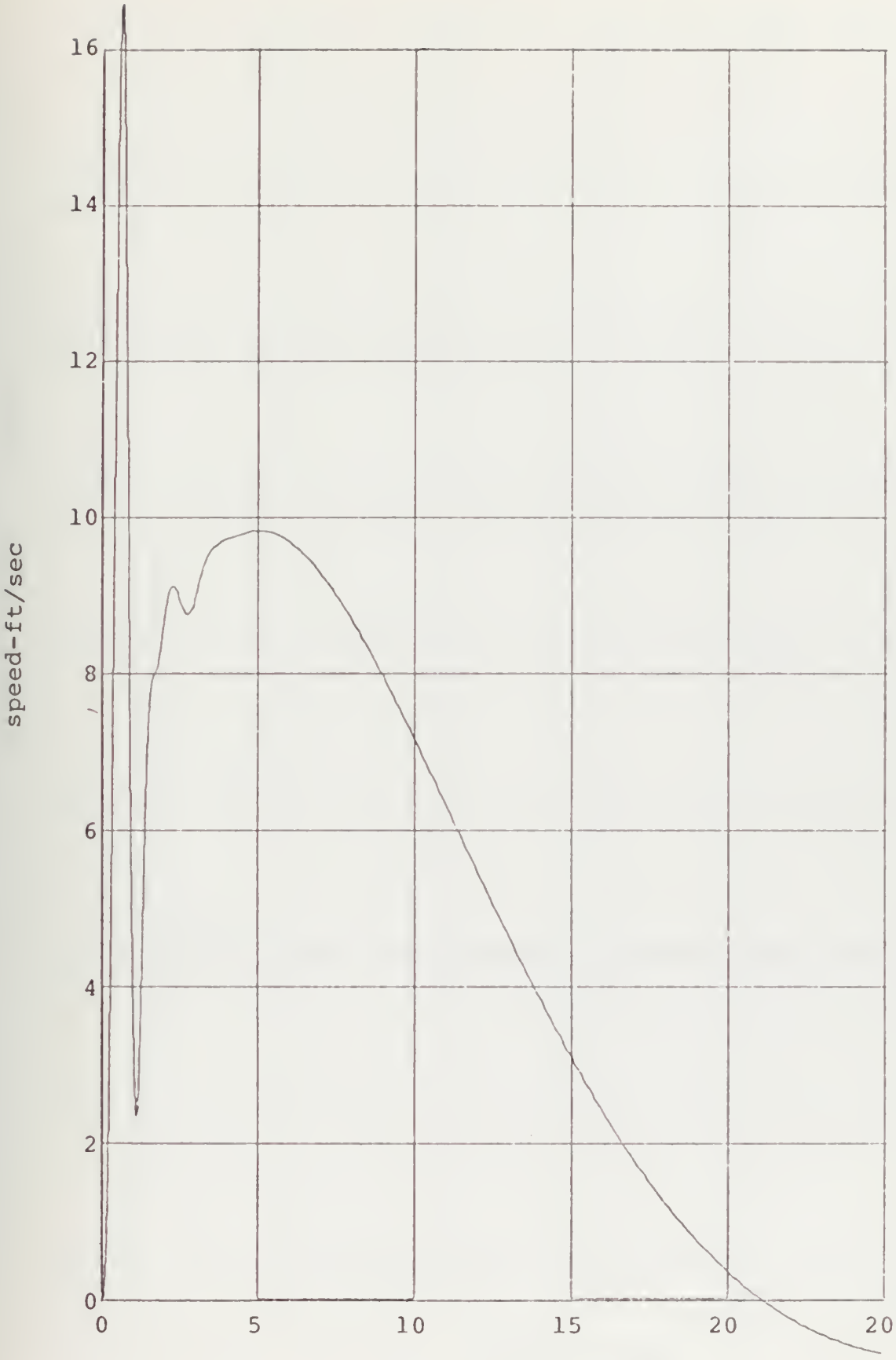


Figure IV - 32

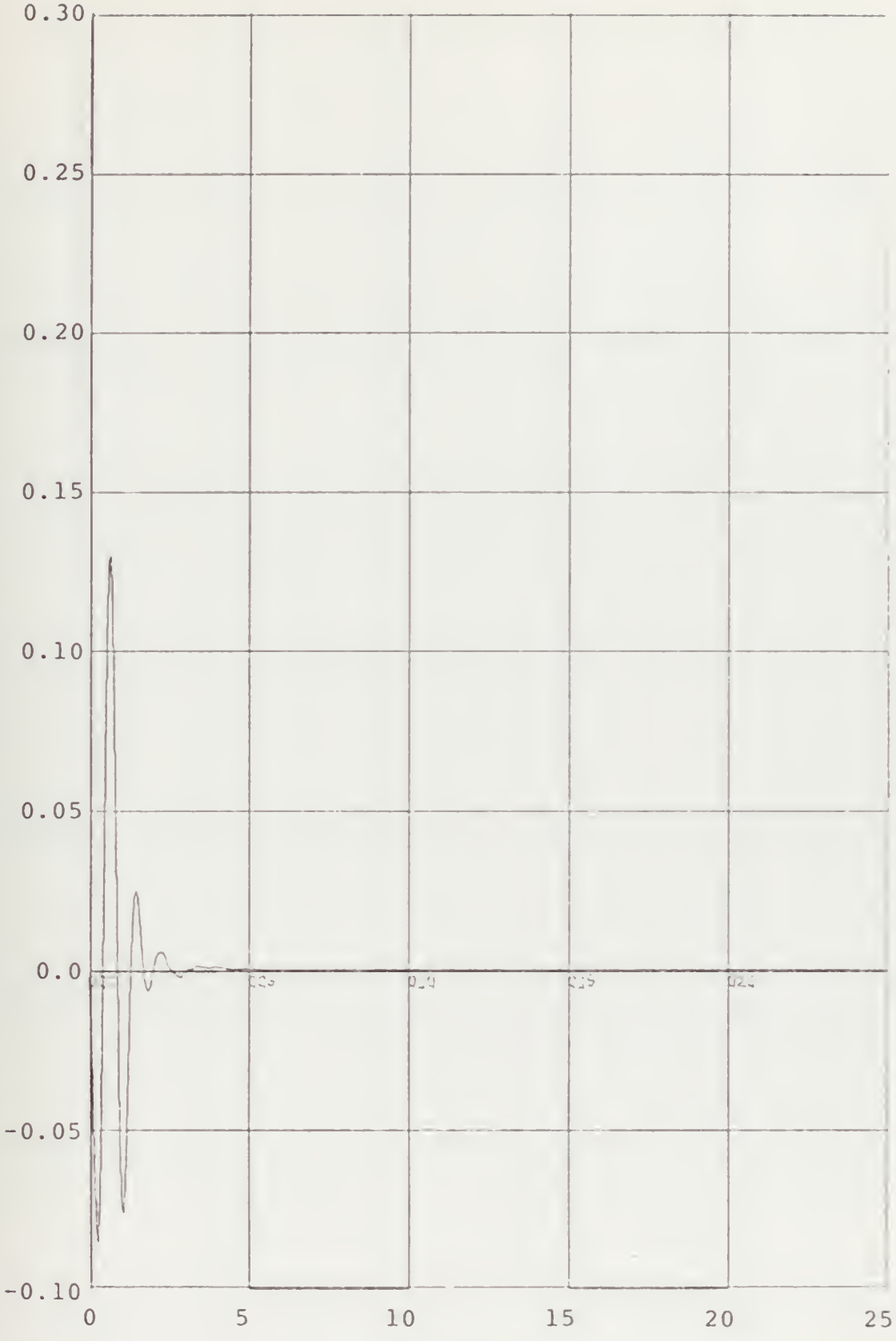

0.30
0.25
0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 33

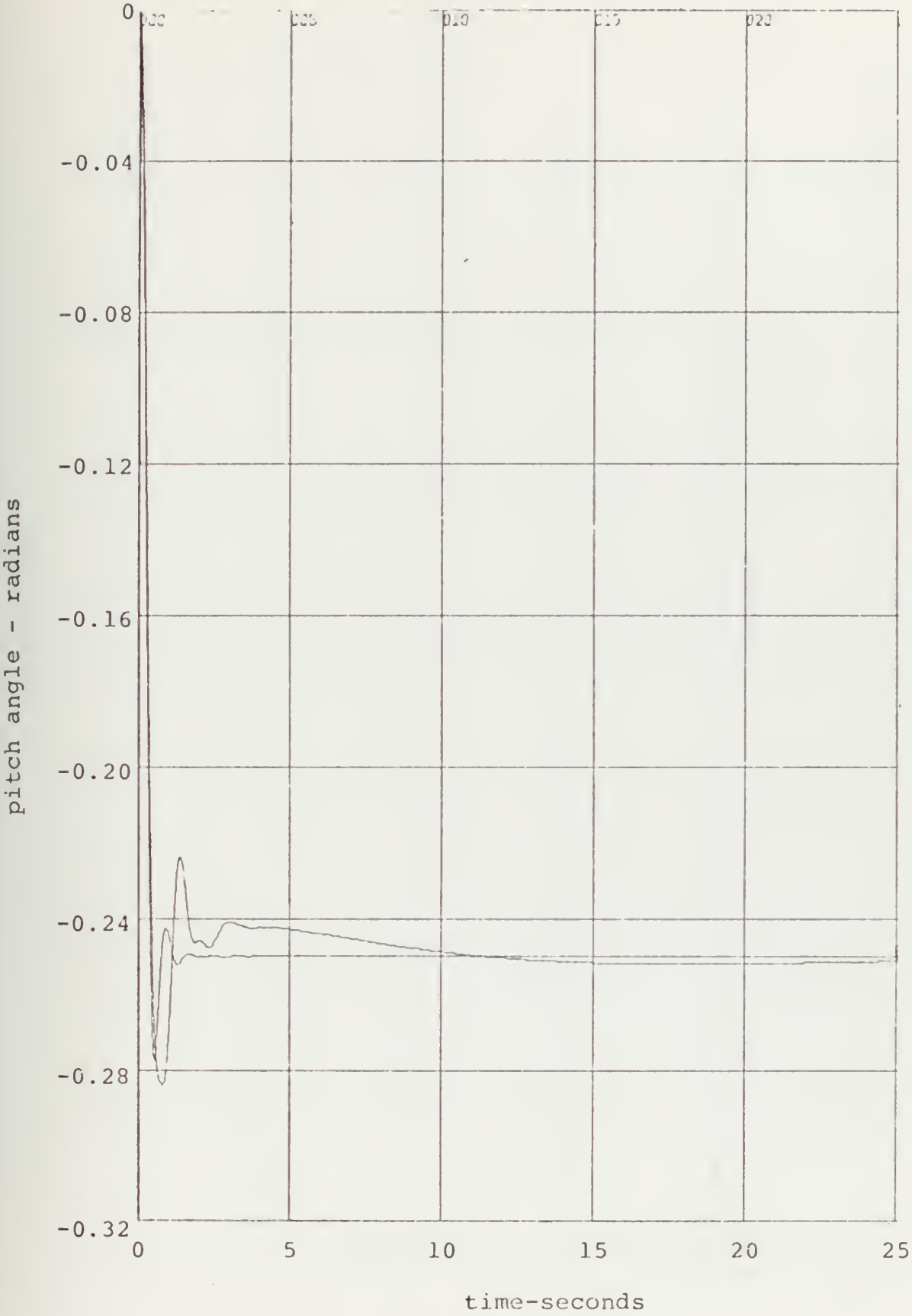

0
-0.04
-0.08
-0.12
-0.16
-0.20
-0.24
-0.28
-0.32
0
5
10
15
20
25
pitch angle - radians
time-seconds

Figure IV - 34

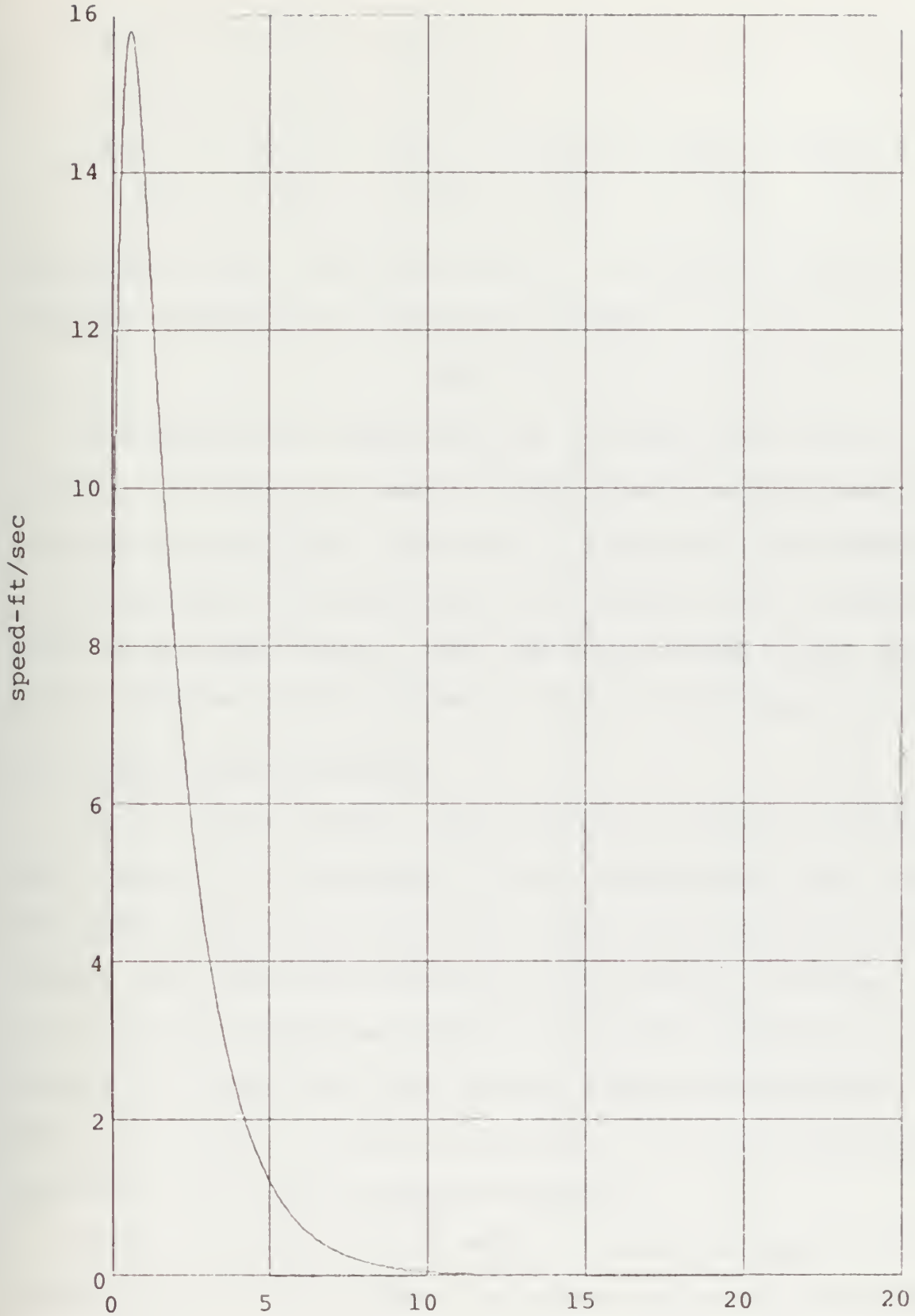


Figure IV - 35

Pole 1	Pole 2	Pole 3			
12.236	22.236	15.354			
Zero 1	Zero 2	Zero 3	Zero 4	Zero 5	Zero 6
1.444	0.302	1.305	0.360	0.436	1.290

Parameters found, arbitrary start, $J = f(u,\theta) = .04667$, arbitrary response BL = 0.25 on all zeros.

Table IV - 11

The results are shown plotted in Figure IV - 36, IV - 37 and IV - 38 for u, α and θ respectively and the excellent results can be noted. This was also the best that BOXPLX could do since it exited after two restarts with no improvement in the best vertex. One need only glance at the original response of the system to note the improvement obtained.

C. SENSITIVITY OF RESULTS

As was noted earlier, some excellent response matching has occured in the process of this investigation where, in the initial phase, the poles and zeros found by BOXPLX varied widely from those which produced the reference response. As a result there could be orders of magnitude difference in values of J and still the results could be acceptable. Thus, of itself, a "small" value for J is not necessarily indicative of a good response matching.

A case in point is that where J was a function of all three outputs. The graphed responses were so superimposed as to be almost indistinguishable yet the value of Pole 1

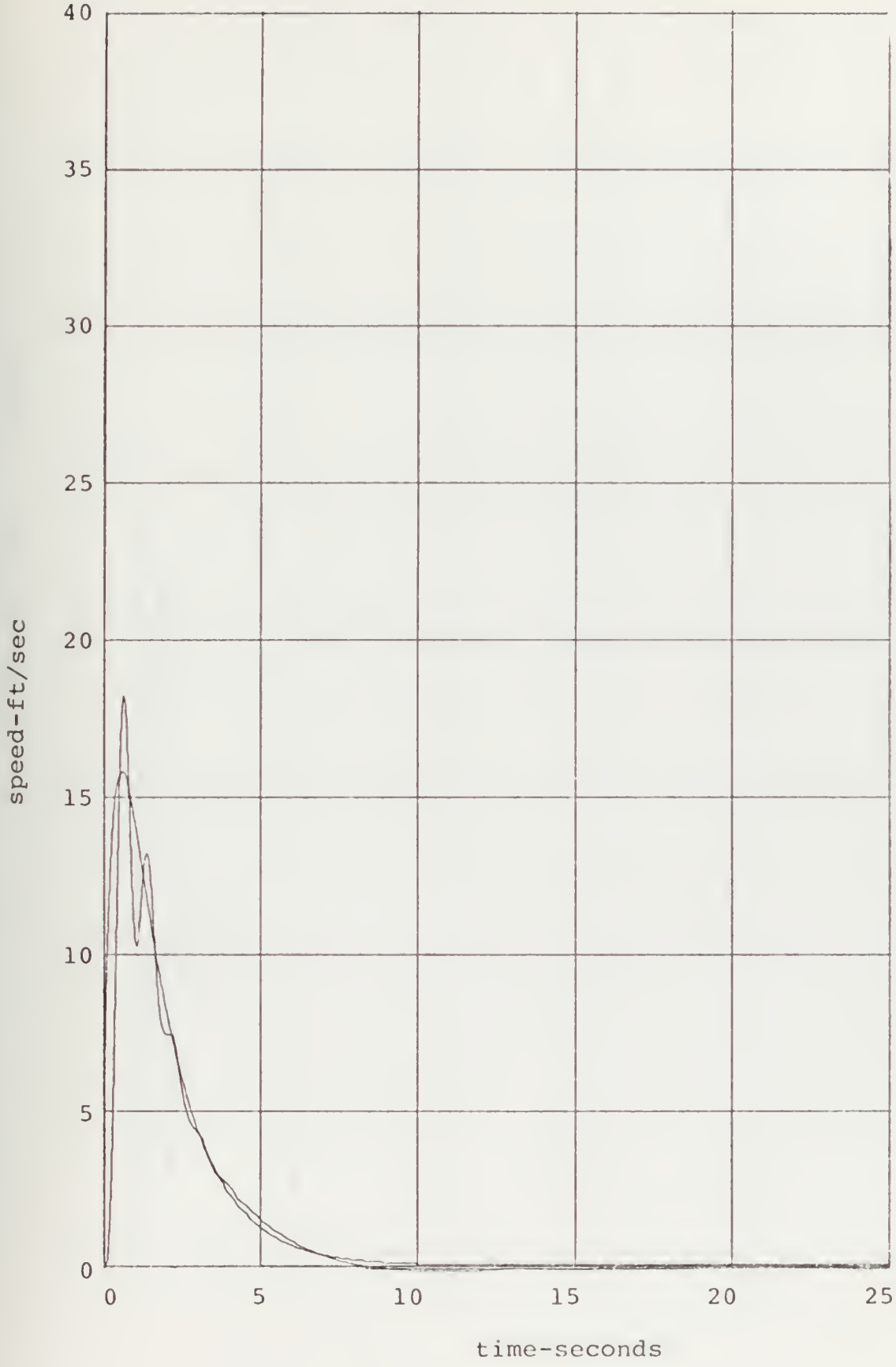


Figure IV - 36

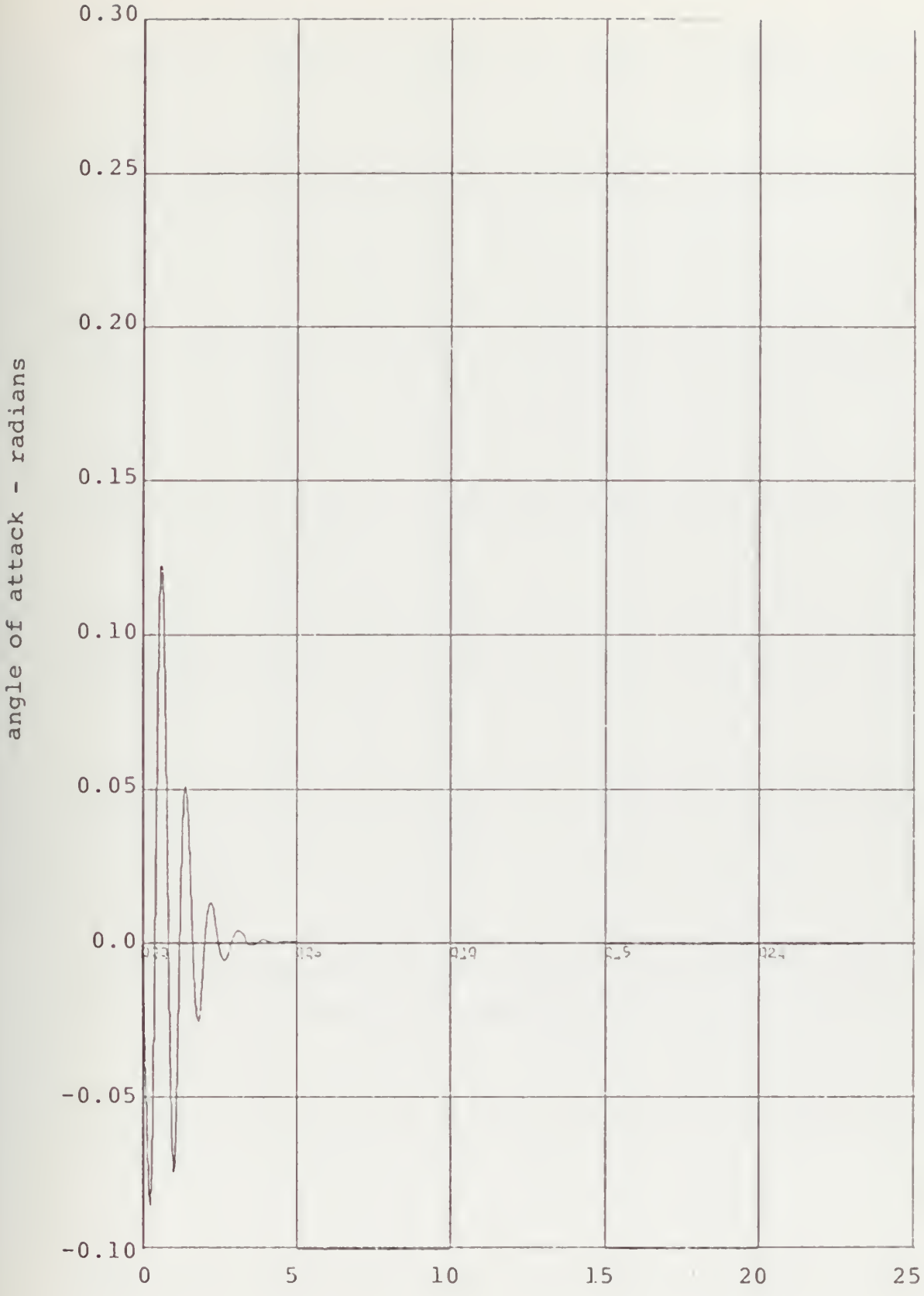

0.30
0.25
0.20
0.15
0.10
0.05
0.0
-0.05
-0.10
angle of attack - radians
0
5
10
15
20
25
time-seconds

Figure IV - 37

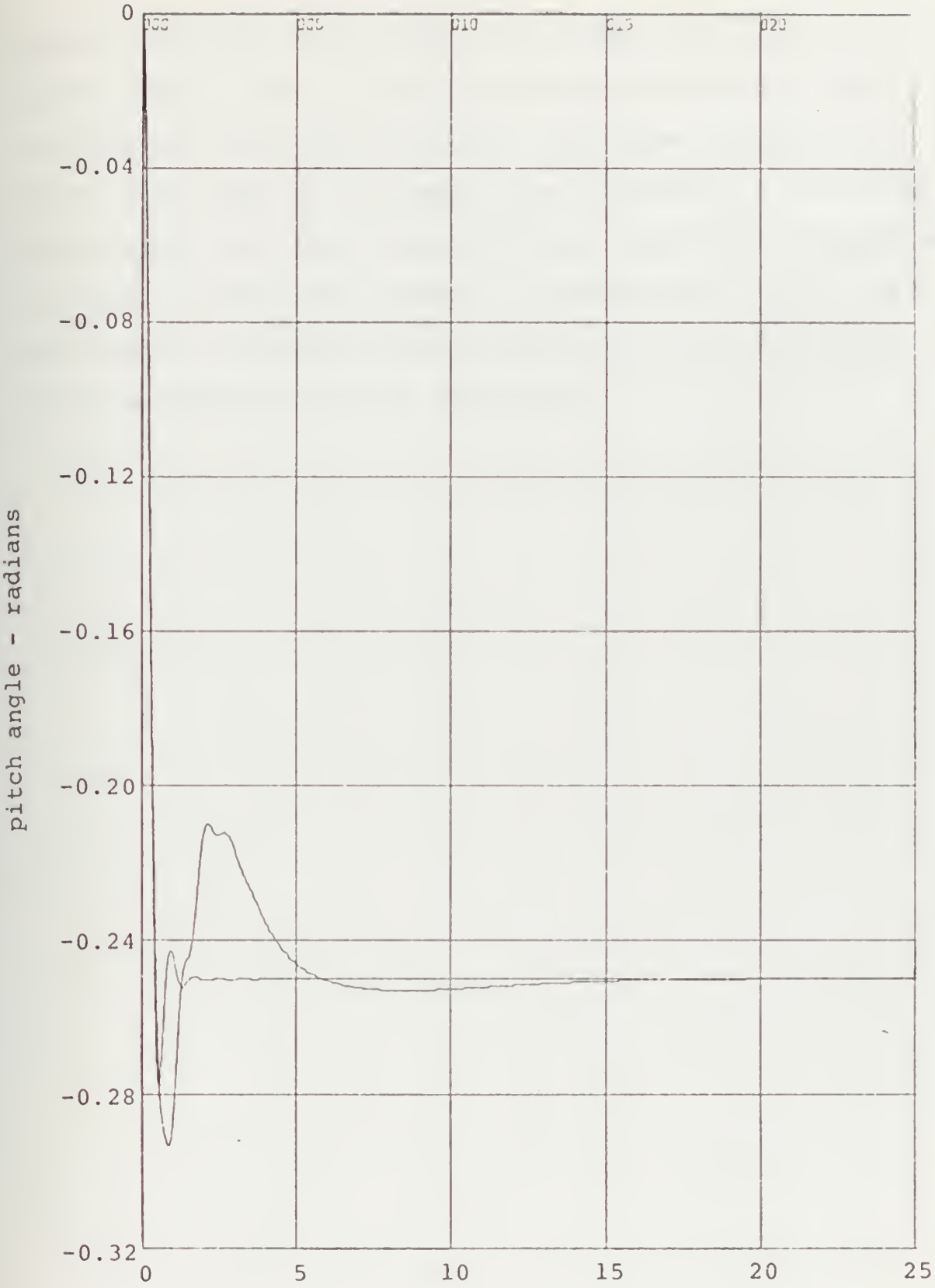

0
-0.04
-0.08
-0.12
-0.16
-0.20
-0.24
-0.28
-0.32
pitch angle - radians
0
5
10
15
20
25
time-seconds

Figure IV - 38

varied from its actual value by over 50% and that for Zero 3 by over 100%. Both of these parameters were in the speed compensator and it might be said that these variations together with those of the other three parameters in the same compensator might have interacted with each other to produce an overall acceptable response. Nonetheless, for all the parameters, the results show insensitivity for variations in the parameters of up to 25 and 30%.

V. CONCLUSIONS & RECOMMENDATIONS FOR FURTHER RESEARCH

A. CONCLUSIONS

The investigations carried out in this thesis result in the following conclusions:

a. It is possible in a multivariable, steady-state decoupled system with cascade compensators and unity feedback to design the compensators by computer methods to have the system match any reasonable specified response to a given input.

b. Where computer time is a problem, there exist several methods, as outlined in this paper, for the function minimization subroutine used here to reduce the time taken to produce an acceptable response match.

c. Where only a limited number of parameters are available, any arbitrary desired response should have time constants compatible with the natural time constants of the system.

d. Where possible, a cost function which is a function of two or more outputs, when balanced, will produce better results than when it is a function of only one output.

e. Where the cost function is a function of only one output, the output to be chosen should be the one which is commanded by the input.

f. The results show insensitivity of response to variations in parameters of up to 25 and 30%.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

The following areas are recommended to extend the use of the design tool developed in this paper:

a. The gains in the compensators, which were kept constant in this investigation, could be made free parameters to extend the flexibility of the design procedure.

b. The numbers of poles and zeros in each compensator were left at those developed in Huang's paper. An investigation should be made to see if the response matching could be improved by increasing the numbers of poles and zeros, and by how many.

c. The flexibility of the method would be further increased if provision were made to include complex poles and zeros as well as real ones. By using transfer functions of the form $\frac{s^2 + as + b}{(s+p_1)(s+p_2)}$, where a and b are real numbers, the parameters would be a combination of poles, zeros and coefficients.

d. Military specifications for aircraft contain specific values for limits on parameters of aircraft motion such as short period and phugoid frequency and damping. If the equations were altered to obtain these parameters of the aircraft response in terms of the compensator poles and zeros, a matching to military specifications could be obtained by insertion of a suitable implicit constraint evaluation subroutine to BOXPLX.

APPENDIX A

DESCRIPTION OF SUBROUTINE BOXPLX

BOXPLX is a Fortran-63 subroutine which finds the minimum of a general non-linear function within a feasible region by the complex method of M.J. Box. Explicit constraints are defined as upper and lower bounds on the independent variables. Implicit constraints may be arbitrary functions of the variables. Two function subprograms to evaluate the cost function and the implicit constraints respectively msut be appended by the user.

The complex method is an extension and adaptation of the simplex method of linear programming. Starting with any one feasible point (XS(I)) in n-dimensional space for n variables, a "complex" of 2n vertices (sets of variables) is constructed by selecting random points (via a random number generator) within the feasible region. For this purpose, n coordinates are first randomly chosen within the space bounded by the explicit constraints. This defines a trial initial vertex, which is then checked for possible violation of the implicit constraints. If one or more are violated, the trial initial vertex is displaced half of its distance from the centroid ofpreviously selected initial vertices. This displacement process continues for up to 5n displacements until the vertex has become feasible. If not, exit occurs from the subroutine.

If an initial complex is established, the basic computational loop is initiated. The current worst vertex, i.e., the one with the largest value of the cost function, is replaced by its over-reflection through the centroid of all other vertices. (If the vertex to be replaced is considered to be a vector in n-space, its over-reflection is opposite in direction, increased in length by a factor 1.3 and collinear with the replaced vertex and the centroid of all other vertices).

When this overreflection is not feasible or remains the worst vertex, it is displaced half-way toward the centroid. If NV displacements occur without a successful result, it is over reflected through the best vertex, i.e., the one with the smallest value of the cost function. If a successful result is still not achieved, the whole process restarts at the best vertex or the centroid, whichever has the smaller cost function.

However, in most cases, all vertices converge to a point which is taken as a solution. Convergence is considered attained when $2n$ consecutive cost function values have been within 10^{-10} of a base value.

The parameter list for BOXPLX is flexible and allows the user to choose:

a. the number of variables
b. the number of auxiliary variables (implicit constraints)
c. the frequency of printed output
d. the number of trials allowed
e. the first random number to be used

f. the starting values
g. the upper and lower bounds

If BOXPLX does not find a minimum cost function at the end of the number of trials specified it prints out the values for the best vertex reached up to that point. If further trials are considered necessary, these values should be used as the starting values for the next run.

Obviously, the starting values selected must lie in the feasible region, i.e., between the upper and lower bounds, or BOXPLX will not even start.

If calculation restarts at the best vertex after being unable to find a feasible vertex and the same situation occurs again, the "new" best vertex is compared with the previous one. If it is better, calculations restart again at this "new" best vertex, otherwise exit from BOXPLX occurs with the previous best vertex being printed out.

ARBITRARY THETA RESPONSE GENERATING PROGRAMME

```
//MACBX1 JOB (1679,0074,AA34),'MACNAMARA SMC 1134'
// EXEC FORTCLGP,REGION.GO=100K
//FORT.SYSIN DD *
C   STATE VARIABLE SIMULATION OF SYSTEM WITH THETA RESPONSE HAVING 10 PERCENT    C
C   MAXIMUM OVERSHOOT AND SETTLING IN 25 SECONDS WITH STEP INPUT OF -0.25       C
C   ARRANGED TO PUNCH CARDS OF OUTPUT AT 0.05 SECOND INTERVALS                  C
      DIMENSION TIME(501),THETA(501)
      REAL*8 X(4),XDOT(4),T,DT,TITLE(12)/12*'        '/
      REAL*4 LABEL/'    '/
      DO 1 I=1,4
      X(I)=0.0
    1 XDOT(I)=0.0
      T=0.0
      DT=0.05
      NT=0
      THETAI=-0.25
      DO 7 I=1,501
    2 XDOT(1)=X(2)
      XDOT(2)=X(3)
      XDOT(3)=X(4)
      XDOT(4)=-5861.8*X(1)-1741.8*X(2)-260.31*X(3)-22.75*X(4)+5861.8*THE
     *TAI
      S=RKLDEQ(4.,X,XDOT,T,DT,NT)
      IF(S-1.0) 4,2,6
    4 WRITE(6,5)
    5 FORMAT(//T8,'INTEGRATION TROUBLE')
      STOP
    6 TT=I
      TIME(I)=TT*DT
      THETA(I)=X(1)
      WRITE(6,10) TIME(I),THETA(I)
   10 FORMAT(2X,1F6.2,1E15.7)
    7 CONTINUE
      WRITE(7,11) THETA
   11 FORMAT(8F10.6)
      CALL DRAW(501,TIME,THETA,0,0,LABEL,TITLE,5.,0,0,0,0,0,6,9,1,L)
      WRITE(6,8) L
    8 FORMAT(I6)
      STOP
      END
```

ARBITRARY SPEED RESPONSE GENERATING PROGRAMME

```
//MACCX1 JOB (1679,0074,AA34),'MACNAMARA SMC 1134'
// EXEC FORTCLGP,REGION.GO=90K
//FORT.SYSIN DD *
C   THIS PROGRAM GENERATES AN ARBITRARY SPEED RESPONSE OF THE AIRCRAFT TO A      C
C   STEP ELEVATOR INPUT OF -0.25 AND PUNCHES CARDS REPRESENTING THAT OUTPUT AT   C
C   0.05 SECOND INTERVALS                                                       C
      DIMENSION TIME(501),SPEED(501)
      REAL*8 X(2),XDOT(2),T,DT,TITLE(12)/12*'          '/
      REAL*4 LABEL/'    '/
      DO 1 I=1,2
      X(I)=0.0
    1 XDOT(I)=0.0
      T=0.0
      DT=0.05
      NT=0
      XIN=1.0
      DO 7 I=1,501
    2 XDOT(1)=X(2)
      XDOT(2)=-1.8496*X(1)-3.536*X(2)+1.8496*XIN
      S=RKLDEQ(2,X,XDOT,T,DT,NT)
      IF(S-1.0) 4,2,6
    4 WRITE(6,5)
    5 FORMAT(//T8,'INTEGRATION TROUBLE')
      STOP
    6 TT=I
      TIME(I)=TT*DT
      SPEED(I)=38.0*X(2)
      WRITE(6,10) TIME(I),SPEED(I)
   10 FORMAT(2X,1F6.2,1E15.7)
    7 CONTINUE
      WRITE(7,11) SPEED
   11 FORMAT(8F10.6)
      CALL DRAW(501,TIME,SPEED,0,0,LABEL,TITLE,5.,2.,0,0,2,2,6,9,1,L)
      WRITE(6,8) L
    8 FORMAT(I6)
      STOP
      END
```

```
                COMPENSATOR POLE AND ZERO LOCATING PROGRAMME
                TO MATCH ARBITRARY THETA AND SPEED RESPONSE
                   J IS A FUNCTION OF PITCH ANGLE & SPEED


//MACBX8 JOB (1679,0074,AA34),'MACNAMARA SMC 1134',TIME=36
//  EXEC FORTCLGP,REGION.GO=125K
//FORT.SYSIN DD *
      EQUIVALENCE(XMN,XS)
      COMMON XDATA1,XDATA2
      DIMENSION XS(9),XMN(9),BU(9),BL(9),XDATA1(501),XDATA2(501)
      REAL*8 X(16),XDOT(16),T,DT
      CALL ERRSET(207,256,25,0,0,206)
      N=501
      NTZ=1500
      R=0.1
C   THE DESIRED AIRCRAFT RESPONSE
      READ(5,5) (XDATA1(I),I=1,N)
      READ(5,5) (XDATA2(I),I=1,N)
    5 FORMAT(8F10.6)
      WRITE(6,6)
    6 FORMAT(//T8,'THE DESIRED A/C RESPONSE TO AN ELEVATOR STEP INPUT
     C AT INTERVAL OF 0.05 SECONDS')
      WRITE(6,7) (XDATA1(I),I=1,N)
      WRITE(6,7) (XDATA2(I),I=1,N)
    7 FORMAT(2X,10F10.6)
      NV=9
      NAV=0
C   THE FREE PARAMETER VECTOR IS XMN(I)
C   THE FREE PARAMETER UPPER BOUNDS ARE BU(I)
      READ(5,5) (BU(I),I=1,NV)
      WRITE(6,5) (BU(I),I=1,NV)
C   THE FREE PARAMETER LOWER BOUNDS ARE BL(I)
      READ(5,5) (BL(I),I=1,NV)
      WRITE(6,5) (BL(I),I=1,NV)
C   THE FREE PARAMETER STARTING VALUES ARE XS(I)
      READ(5,5) (XS(I),I=1,NV)
      WRITE(6,5) (XS(I),I=1,NV)
C   NOW CALL THE FUNCTION MINIMIZATION SUBROUTINE WHOSE MAIN OUTPUTS ARE
C     XMN: THE MINIMUM FOUND VALUES OF THE FREE PARAMETERS
C    YMN: THE VALUE OF THE PERFORMANCE INDEX AT THE MINIMUM
      CALL BOXPLX(NV,NAV,50,NTZ,R,XS,BU,BL,YMN,IER)
      WRITE (6,2) YMN
    2 FORMAT(//T8,'THE MINIMUM PERFORMANCE INDEX =',F10.5)
      WRITE(6,3) XMN(1),XMN(2),XMN(3),XMN(4),XMN(5),XMN(6),XMN(7),XMN(8)
     *,XMN(9)
    3 FORMAT(//,2X,'POLE 1=',F10.5,2X,'POLE 2=',F10.5,2X,'POLE 3=',F10.5
```

```
     *,//,2X,'ZERO 1=',F10.5,2X,'ZERO 2=',F10.5,2X,'ZERO 3=',F10.5,2X,'Z
     *ERO 4=',F10.5,2X,'ZERO 5=',F10.5,2X,'ZERO 6=',F10.5//)
      CALL GRAPH(XMN,XDATA1,XDATA2)
      STOP
      END
```

THE FUNCTION EVALUATION FUNCTION SUBROUTINE

```
      FUNCTION FE(PA)
C   CALCULATES THE VALUE OF THE PERFORMANCE INDEX
      COMMON XDATA1,XDATA2
      DIMENSION PA(9),XDATA1(501),XDATA2(501)
      REAL*8 X(16),XDOT(15),T,DT
C   SET INITIAL CONDITIONS TO ZERO
      DO 1 I=1,16
    1 X(I)=0.0
      T=0.0
      DT=0.05
      NT=0
      R1=0.0
      R2=0.0
      R3=-0.25
      A11=PA(1)+PA(2)
      A12=PA(1)*PA(2)
      A31=PA(3)
      B10=1000.0
      B11=1000.0*(PA(4)+PA(5)+PA(6))
      B12=1000.0*(PA(4)*PA(5)+PA(5)*PA(6)+PA(4)*PA(6))
      B13=1000.0*PA(4)*PA(5)*PA(6)
      B20=-400.0
      B21=-400.0*PA(7)
      B30=-20.0
      B31=-20.0*(PA(8)+PA(9))
      B32=-20.0*PA(8)*PA(9)
      BETA10=B10
      BETA11=B11-(BETA10*A11)
      BETA12=B12-(BETA11*A11)-(BETA10*A12)
      BETA13=B13-(BETA12*A11)-(BETA11*A12)
      BETA20=B20
      BETA21=B21
      BETA30=B30
      BETA31=B31-(BETA30*A31)
      BETA32=B32-(BETA31*A31)
      DO 7 I=1,250
    2 XDOT(1)=-.067*X(1)+22.2*X(2)-32.2*X(4)-6.12*X(7)+.067*X(5)
```

```
      E1=R1-X(1)
      XDOT(2)=-.004017*X(1)-.7263*X(2)+.9479*X(3)-.001355*X(5)-.1527*X(7
     *)-.0752*X(8)
      E2=R2-X(2)
      XDOT(3)=.003755*X(1)-1.826*X(2)-1.452*X(3)-.001732*X(5)+.09953*X(7
     *)-1.945*X(8)
      XDOT(4)=X(3)
      E3=R3-X(4)
      XDOT(5)=X(6)
      XDOT(6)=-1.21*X(5)-1.98*X(6)+1.21*X(10)+1210.0*E1
      XDOT(7)=-X(7)+X(13)-400.0*E2
      XDOT(8)=X(9)
      XDOT(9)=-121.0*X(8)-6.6*X(9)+121.0*X(14)-2420.0*E3
      XDOT(10)=X(11)+BETA11*E1
      XDOT(11)=X(12)+BETA12*E1
      XDOT(12)=-A12*X(11)-A11*X(12)+BETA13*E1
      XDOT(13)=BETA21*E2
      XDOT(14)=X(15)+BETA31*E3
      XDOT(15)=-A31*X(15)+BETA32*E3
      S=RKLDEQ(15,X,XDOT,T,DT,NT)
      IF(S-1.0) 4,2,8
    4 WRITE(6,5)
    5 FORMAT(//T8,'INTEGRATION TROUBLE')
      RETURN
    8 X(16)=X(16)+0.001*(X(1)-XDATA1(I))**2+(X(4)-XDATA2(I))**2
C   THE COMPUTED PERFORMANCE INDEX
      PE=X(16)*0.05
    7 CONTINUE
      RETURN
      END
```

THE IMPLICIT CONSTRAINT EVALUATION FUNCTION SUBROUTINE

```
C   CALCULATES IMPLICIT CONSTRAINTS
      FUNCTION KE(X)
      DIMENSION X(9)
      KE=0
      RETURN
      END
```

THE SIMULATION AND PLOTTING SUBROUTINE

```
C   PLOTS THE RESPONSE OF THE CONPENSATED SYSTEM AND THE DESIRED RESPONSE
      SUBROUTINE GRAPH(PA,XDATA1,XDATA2)
      DIMENSION PA(9),TIME(501),SPEED(501),DSPEED(501),XDATA1(501),ALPHA
     *(501),THETA(501),DTHETA(501),XDATA2(501)
      REAL*8 X(15),XDOT(15),T,DT,TITLE(12)/'MACNAMAR','A M.A.S.',10*'
     *      '/
      REAL*4 LABL1/'DESD'/,LABL2/'ACTL'/
      WRITE(6,9) (PA(I),I=1,9)
    9 FORMAT(2X,10F10.6)
      T=0.0
      DT=0.05
      NT=0
C   SET INITIAL CONDITIONS TO ZERO
      DO 1 I=1,15
      X(I)=0.0
    1 XDOT(I)=0.0
      R1=0.0
      R2=0.0
      R3=-0.25
      A11=PA(1)+PA(2)
      A12=PA(1)*PA(2)
      A31=PA(3)
      B10=1000.0
      B11=1000.0*(PA(4)+PA(5)+PA(6))
      B12=1000.0*(PA(4)*PA(5)+PA(5)*PA(6)+PA(4)*PA(6))
      B13=1000.0*PA(4)*PA(5)*PA(6)
      B20=-400.0
      B21=-400.0*PA(7)
      B30=-20.0
      B31=-20.0*(PA(8)+PA(9))
      B32=-20.0*PA(8)*PA(9)
      BETA10=B10
      BETA11=B11-(BETA10*A11)
      BETA12=B12-(BETA11*A11)-(BETA10*A12)
      BETA13=B13-(BETA12*A11)-(BETA11*A12)
      BETA20=B20
      BETA21=B21
      BETA30=B30
      BETA31=B31-(BETA30*A31)
      BETA32=B32-(BETA31*A31)
      DO 7 I=1,501
    2 XDOT(1)=-.067*X(1)+22.2*X(2)-32.2*X(4)-6.12*X(7)+.067*X(5)
      E1=R1-X(1)
      XDOT(2)=-.004017*X(1)-.7263*X(2)+.9479*X(3)-.001355*X(5)-.1527*X(7
     *)-.0752*X(8)
      E2=R2-X(2)
      XDOT(3)=.003755*X(1)-1.826*X(2)-1.452*X(3)-.001732*X(5)+.09953*X(7
     *)-1.945*X(8)
```

```
      XDOT(4)=X(3)
      E3=R3-X(4)
      XDOT(5)=X(6)
      XDOT(6)=-1.21*X(5)-1.98*X(6)+1.21*X(10)+1210.0*E1
      XDOT(7)=-X(7)+X(13)-400.0*E2
      XDOT(8)=X(9)
      XDOT(9)=-121.0*X(8)-6.6*X(9)+121.0*X(14)-2420.0*E3
      XDOT(10)=X(11)+BETA11*E1
      XDOT(11)=X(12)+BETA12*E1
      XDOT(12)=-A12*X(11)-A11*X(12)+BETA13*E1
      XDOT(13)=BETA21*E2
      XDOT(14)=X(15)+BETA31*E3
      XDOT(15)=-A31*X(15)+BETA32*E3
      S=RKLDEQ(15,X,XDOT,T,DT,NT)
      IF(S-1.0) 4,2,6
    4 WRITE(6,5)
    5 FORMAT(//T8,'INTEGRATION TROUBLE')
      RETURN
    6 TT=I
      TIME(I)=TT*DT
      SPEED(I)=X(1)
      DSPEED(I)=XDATA1(I)
      ALPHA(I)=X(2)
      THETA(I)=X(4)
      DTHETA(I)=XDATA2(I)
      WRITE(6,10) TIME(I),DSPEED(I),SPEED(I),ALPHA(I),DTHETA(I),THETA(I)
   10 FORMAT(2X,1F6.2,5E15.7)
    7 CONTINUE
C     PLOT OF OPTIMUM COMPENSATED THETA RESPONSE  C
      CALL DRAW(501,TIME,THETA,1,0,LABL2,TITLE,5.0,.04,8,0,2,0,6,9,1,L)
      WRITE(6,8) L
C     PLOT OF DESIRED SYSTEM THETA RESPONSE   C
      CALL DRAW(501,TIME,DTHETA,3,0,LABL1,TITLE,5.0,.04,8,0,2,0,6,9,1,L)
      WRITE(6,8) L
    8 FORMAT(I6)
C     PLOT OF OPTIMUM COMPENSATED SPEED RESPONSE  C
      CALL DRAW(501,TIME,SPEED,1,0,LABL2,TITLE,5.0,0,0,0,0,0,6,9,1,L)
      WRITE(6,8) L
C     PLOT OF DESIRED SYSTEM SPEED RESPONSE   C
      CALL DRAW(501,TIME,DSPEED,3,0,LABL1,TITLE,5.0,0,0,0,0,0,6,9,1,L)
      WRITE(6,8) L
C     PLOT OF ALPHA RESPONSE
      CALL DRAW(501,TIME,ALPHA,0,0,LABL2,TITLE,5.0,0,0,0,0,0,6,9,1,L)
      WRITE(6,8) L
      RETURN
      END
```

THE FUNCTION MINIMIZATION SUBROUTINE

```
      SUBROUTINE BOXPLX (NV,NAV,NPR,NTZ,RZ,XS,BU,BL,YMN,IER)
C                                                                       20
      DIMENSION V(50,50), FUN(50), SUM(25), CEN(25), XS(NV), BU(NV), BL(
     1NV)
C                                                                       60
C   IF KKON=0, EXPLICIT CONSTRAINTS ARE NOT APPLIED AFTER INITIAL       70
C               COMPLEX  IS FORMED                                      80
C                                                                       90
      KV = 9
      KV = 100
      KV = 5
      KKON=0
      KKON = 1                                                          100
      EP = 1.E-6
      IF (NTZ .GT. 0)  GO TO 2
      NTA = 2000
      GO TO 3                                                           150
    2 NTA = NTZ                                                         160
    3 R = RZ                                                            170
      IF (R .LE. 0. .OR. R .GE. 1.)  R=1./3.
      NVT = NV+NAV                                                      220
C           TOTAL VARS, EXPLICIT PLUS IMPLICIT                          230
      NT = 0                                                            240
C           CURRENT TRIAL NO.                                           250
      NPT = 0                                                           260
C           CURRENT NO. OF PERMISSIBLE TRIALS                           270
      NTFS = 0                                                          280
C           CURRENT NO. OF TIMES F HAS BEEN ALMOST UNCHANGED            290
C                                                                       300
C             CHECK FEASIBILITY OF START POINT                          310
C                                                                       320
      DO 13 I=1,NV                                                      330
      VT = XS(I)                                                        360
      IF (BL(I) .LE. VT)  GO TO 8
      II = -I
      VT = BL(I)                                                        390
      GO TO 10                                                          400
    8 IF (BU(I) .GE. VT)  GO TO 12
      II = I
      VT = BU(I)                                                        430
   10 IF (NPR .GT. 0)  WRITE (6,11) II
   11 FORMAT (50HOINDEX AND DIRECTION OF OUTLYING VARIABLE AT STARTI5)  450
   12 V(I,1) = VT                                                       460
      CEN(I) = VT                                                       470
      BL(I) = BL(I)+AMAX1(EP,EP*ABS(BL(I)))                            480
```

```
      BU(I) = BU(I)-AMAX1(EP,EP*ABS(BU(I)))                              490
   13 SUM(I) = VT                                                        500
      NCE = 1                                                            530
C           NUMBER OF CONSTRAINT EVALUATIONS                             540
      I = 1                                                              550
      IF (KE(V(1,1)) .EQ. 0)  GO TO 17
      IF (NPR .LE. 0)  GO TO 25
      WRITE (6,16)
   16 FORMAT (50H0IMPLICIT CONSTRAINT VIOLATED AT START. DEAD END. )     590
      GO TO 25                                                           600
   17 NFE = 1                                                            610
C
C    NUMBER OF VERTICES (K) = 2 TIMES NO. OF VARIABLES.
      K=NV
      K = 2*NV                                                           660
C
C    NUMBER OF DISPLACEMENTS ALLOWED.
      NLIM = 20
      ALPHA = 1.3                                                        670
      FK = K                                                             680
      FKM = FK-1.                                                        690
      BETA = ALPHA+1.                                                    700
C
C    INSURE SEED OF RANDOM NUMBER GENERATOR IS ODD.
      IQR = R*1.E7                                                       710
      IF (MOD(IQR,2).EQ.0) IQR=IQR+101                                   720
C                                                                        730
C                        SET UP INITIAL VERTICES                         740
      FUN(1) = FE(V(1,1))
      YMN = FUN(1)
   18 FI = 1.                                                            760
      FUNOLD = FUN(1)                                                    770
C                                                                        780
      DO 28 I=2,K                                                        790
      FI = FI+1.                                                         800
C                                                                        810
      DO 19 J=1,NV                                                       820
C
C    RANDOM NUMBER GENERATOR  (RANDU)
      IQR = IQR *65539
      IF (IQR .LT. 0)  IQR=IQR +2147483647 +1
      RQX = IQR
      RQX = RQX *.4656613E-9
   19 V(J,I) = BL(J)+RQX*(BU(J)-BL(J))                                   840
C                                                                        850
      DO 22 L=1,NLIM                                                     880
      NCE = NCE+1                                                        890
      IF (KE(V(1,I)) .EQ. 0)  GO TO 26
```

```
      DO 21 J=1,NV
   21 V(J,I) = .5*(V(J,I)+CEN(J))                                        930
C                                                                        940
   22 CONTINUE                                                           950
C                                                                        960
      IF (NPR .LE. 0)  GO TO 25
      WRITE (6,24) I
   24 FORMAT (22HOCANNOT FIND FEASIBLE I4,19HTH VERTEX AT START )        990
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,I,FUN,CEN,I)
   25 IER = -1
      GO TO 101
C                                                                       1030
   26 DO 27 J=1,NV                                                      1040
      SUM(J) = SUM(J)+V(J,I)                                            1050
   27 CEN(J) = SUM(J)/FI                                                1060
C                                                                       1070
      NFE = NFE+1                                                       1080
      FUN(I) = FE(V(1,I))                                               1090
   28 CONTINUE                                                          1100
C                                                                       1120
C    END OF LOOP SETTING OF INITIAL COMPLEX.
      IF (NPR .LE. 0)  GO TO 30
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,0)
      J = 1
C
      DO 29 I=2,K
      IF (FUN(J) .GE. FUN(I))  GO TO 29
      J = I
   29 CONTINUE
C                                                                       1180
C    BASIC LOOP.  ELIMINATE EACH WORST VERTEX IN TURN.  IT MUST BECOME
C    NO LONGER WORST, NOT MERELY IMPROVED.  FIND NEXT-TO-WORST VERTEX,
C    THE 'JN'TH ONE.
   30 JN = 1
      IF (J .EQ. 1)  JN =2
      DO 240 I=1,K
      IF (I .EQ. J)  GO TO 240
      IF (FUN(JN) .GE. FUN(I))  GO TO 240
      JN = I
  240 CONTINUE
C                                                                       1200
C    LIMT = NUMBER OF MOVES DURING THIS TRIAL TOWARD THE CENTROID
C     DUE TO FUNCTION VALUE.
      LIMT = 1
C                                                                       1270
C    COMPUTE CENTROID AND OVER REFLECT WORST VERTEX.
      DO 32 I=1,NV                                                      1280
      VT = V(I,J)
```

```
      SUM(I) = SUM(I)-VT                                                1310
      CEN(I) = SUM(I)/FKM                                               1320
   32 V(I,J) = BETA*CEN(I)-ALPHA*VT
C                                                                       1340
      NT = NT+1                                                         1350
C           TEST FIRST FOR CONSTRAINT VIOLATION                         1360
   33 IF (KKON .LE. 0)  GO TO 36
C                                                                       1380
C    INSURE THE EXPLICIT CONSTRAINTS ARE OBSERVED.
      DO 35 I=1,NV
      VT = AMIN1(V(I,J),BU(I))
   35 V(I,J) = AMAX1(VT,BL(I))
C                                                                       1430
C                                                                       1440
C    CHECK FOR IMPLICIT CONSTRAINT VIOLATION.
   36 DO 39 N=1,NLIM                                                    1450
      NCE = NCE+1                                                       1460
      IF (KE(V(I,J)) .EQ. 0)  GO TO 53
C
C    EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE
C    BEST VERTEX.
      IF (MOD(N,KV) .NE. 0)  GO TO 37
      CALL FBV (K,FUN,M)
      DO 360I=1,NV
  360 V(I,J) = BETA*V(I,M) -ALPHA*V(I,J)
      GO TO 370
C                                                                       1480
C    CONSTRAINT VIOLATION:  MOVE NEW POINT TOWARD CENTROID.
   37 DO 38 I=1,NV
   38 V(I,J) = .5*(CEN(I)+V(I,J))
C                                                                       1520
  370 NT = NT+1                                                         1530
   39 CONTINUE                                                          1540
      IER = 1
C                                                                       1550
C    CANNOT GET FEASIBLE VERTEX BY MOVING TOWARD CENTROID,
C    OR CANNOT MAKE THE OBJECTIVE FUNCTION OF THE TRIAL VERTEX NO LONGER
C    THE WORST.
   40 IF (NPR .LE. 0)  GO TO 100
      WRITE (6,42) NT,J
   42 FORMAT (10HOAT TRIAL I4,54H CANNOT FIND FEASIBLE VERTEX WHICH IS N
     1O LONGER WORST,I4,15X,'RESTART FROM BEST VERTEX.')
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,J)
      GO TO 100
C                                                                       1880
C    FEASIBLE VERTEX FOUND, IS IT NO LONGER WORST?
   53 NFE = NFE+1                                                       1890
      FUNTRY = FE(V(I,J))
```

```
C                                                                       1910
C    TEST TO SEE IF FUNCTION VALUE HAS NOT CHANGED.
       AFO = ABS (FUNTRY -FUNOLD)
       AMX = AMAX1 (ABS (EP *FUNOLD), EP)
C ********/*/*/*/*/*/*/*/*/*    TEMP OUTPUT  */*/*/*/*/*/*
C      WRITE (6,99) J,AFO,AMX,FUNTRY,FUNOLD,FUN(J),FUN(JN),NTFS
       WRITE (6,99) J,AFO,AMX,FUNTRY,FUNOLD,FUN(J),FUN(JN),NTFS
    99 FORMAT (1X,I3,6E15.7,I5)
       IF (AFO .GT. AMX)  GO TO 55
       NTFS = NTFS+1
       IF (NTFS .LT. K)  GO TO 56
       IER = 0
       IF (NPR .LE. 0)  GO TO 100
       WRITE (6,80) K
    80 FORMAT (40HOFUNCTION HAS BEEN ALMOST UNCHANGED FOR I5,7H TRIALS)     2530
       GO TO 100
    55 NTFS = 0                                                             1950
    56 IF (FUNTRY .LT. FUN(JN))  GO TO 60
C                                                                       1970
C    TRIAL VERTEX IS STILL WORST; ADJUST TOWARD CENTROID.
C    EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE
C    BEST VERTEX.
       LIMT = LIMT +1
       IF (MOD(LIMT,KV) .NE. 0)  GO TO 59
       CALL FBV (K,FUN,M)
       DO 591 I=1,NV
       VT=BETA*V(I,M)-ALPHA*V(I,J)
   591 V(I,J)=AMIN1(AMAX1(VT,BL(I)),BU(I))
       GO TO 590
    59 DO 58 I=1,NV
    58 V(I,J) = .5*(CEN(I)+V(I,J))
C                                                                       2010
   590 IF (LIMT .LT. NLIM)  GO TO 580
       IER = 2
       IF (NPR .GT. 0)  WRITE (6,42) NT,J
       GO TO 100
   580 NT = NT+1
       GO TO 33                                                         2050
C                                                                       2060
C    SUCCESS:  WE HAVE A REPLACEMENT FOR VERTEX J.
    60 FUN(J) = FUNTRY
       FUNOLD = FUNTRY                                                  2080
       NPT = NPT+1                                                      2090
C                                                                       2100
       DO 61 I=1,NV                                                     2110
    61 SUM(I) = SUM(I)+V(I,J)
C                                                                       2150
       IF (NPR .LE. 0)  GO TO 650
```

```
      IF (MOD (NPT,NPR) .NE. 0)  GO TO 650
C                                                                       2200
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,J)
C
C   HAS THE MAX. NUMBER OF TRIALS BEEN REACHED WITHOUT CONVERGENCE?
C   IF NOT, GO TO NEW TRIAL.
  650 IF (NT .GE. NTA)  GO TO 651
C
C   NEXT-TO-WORST VERTEX NOW BECOMES WORST.
      J = JN
      GO TO 30
  651 IER = 3
      IF (NPR .GT. 0)  WRITE (6,68)
   68 FORMAT (27HOLIMIT ON TRIALS EXCEEDED. )                           2280
C
C   COLLECTOR POINT FOR ALL ENDINGS.
C    1)  CANNOT DEVELOP FEASIBLE VERTEX.            IER = 1
C    2)  CANNOT DEVELOP A NO-LONGER-WORST VERTEX.   IER = 2
C    3)  FUNCTION VALUE UNCHANGED FOR K TRIALS.     IER = 0
C    4)  LIMIT ON TRIALS REACHED.                   IER = 3
C    5)  CANNOT FIND FEASIBLE VERTEX AT START.  (IN THIS CASE OTHER
C        PARAMETERS ARE MEANINGLESS.)               IER = -1
  100 CONTINUE
C
C   FIND BEST VERTEX.
      CALL FBV (K,FUN,M)
      DO 687 I=1,NV
      SUM(I)=0.0
      DO 688 L=1,K
      SUM(I)=SUM(I)+V(I,L)
  688 CONTINUE
  687 CEN(I)=SUM(I)/FK
      FCEN=FE(CEN)
      IF(FUN(M).LE.FCEN) GO TO 692
      FUN(M)=FCEN
      DO 686 I=1,NV
  686 V(I,M)=CEN(I)
  692 IF (IER .GE. 3)  GO TO 103
C
C   RESTART IF THIS SOLUTION IS SIGNIFICANTLY BETTER THAN THE PREVIOUS,
C   OR IF THIS IS THE FIRST TRY.
C   *****/*/*/*/   TEMP   OUTPUT      /*/*/*/*/*/*/*/*/*/*
      WRITE (6,99) (M,YMN,FUN(M))
      IF (FUN(M) .GE. YMN)  GO TO 102
      IF (ABS(FUN(M) -YMN) .LE. AMAX1(EP,EP*YMN))  GO TO 102
C
C   GIVE IT ANOTHER TRY UNLESS LIMIT ON TRIALS REACHED.
  103 YMN = FUN(M)
```

```
      FUN(1) = FUN(M)
      DO 694 I=1,NV
      XS(I) = V(I,M)
      CEN(I) = V(I,M)
      SUM(I) = V(I,M)
  694 V(I,1) = V(I,M)
      IF (IER .LT. 3)  GO TO 18
  102 IF (NPR .LE. 0)  GO TO 101
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,V(1,M),-1)
  101 RETURN
      END                                                              2560
      SUBROUTINE FBV (K,FUN,M)
      DIMENSION FUN(50)
      M = 1
      DO 1 I=2,K
      IF (FUN(M) .LE. FUN(I))  GO TO 1
      M = I
    1 CONTINUE
      RETURN
      END
      SUBROUTINE BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FN,C,IK)
      DIMENSION V(50,50), FN(50), C(25)                                  20
      WRITE (6,1) NT,NPT,NFE,NCE                                         30
    1 FORMAT ('ONO. TOTAL TRIALS = ',I5,4X,'NO. FEASIBLE TRIALS = ',
     XI5,4X,'NO. FUNCTION EVALUATIONS = ',I5,4X,'NO. CONSTRAINT EVALUATI
     XONS = ',I5/'0     FUNCTION VALUE',6X,'INDEPENDENT VARIABLES/DEPEND
     XENT OR IMPLICIT CONSTRAINTS')
C                                                                        70
C                                                                        80
      DO 3 I=1,K                                                         90
      WRITE (6,2) FN(I),(V(J,I),J=1,NV)                                 100
    2 FORMAT (1H ,E18.7,2X,7E14.7/(21X,7E14.7))                         110
      IF (NVT .LE. NV)  GO TO 3
      NVP = NV+1                                                        120
      WRITE (6,4) (V(J,I),J=NVP,NVT)
    3 CONTINUE
C                                                                       140
    4 FORMAT (21X,7E14.7)                                               150
      IF (IK .NE. 0)  GO TO 7
C                                                                       160
    6 WRITE (6,5) (C(I),I=1,NV)                                         170
    5 FORMAT (10HOCENTROID 11X,7E14.7/(21X,7E14.7))                     180
      RETURN                                                            190
    7 IF (IK .GE. 0)  GO TO 8
      WRITE (6,11) (C(I),I=1,NV)
   11 FORMAT ('0  BEST VERTEX',9X,7E14.7/(22X,7E14.7))
      RETURN
    8 WRITE (6,12) IK,(C(I),I=1,NV)
```

```
   12 FORMAT ('0CENTROID LESS VRT ',I2,2X,7E14.7/(22X,7E14.7))
      RETURN
      END                                                              200


C     INPUT DATA FOR SPEED,THETA,BU,BL,XS

//GO.SYSIN DD *
  3.218611  5.902002  8.125541  9.954325 11.444577 12.644844 13.597034 14.337314
 14.896887 15.302668 15.577859 15.742456 15.813685 15.806375 15.733282 15.605378
 15.432087 15.221498 14.980546 14.715174 14.430464 14.130754 13.819745 13.500587
 13.175952 12.848103 12.518950 12.190098 11.862891 11.538451 11.217703 10.901406
 10.590181 10.284524  9.984827  9.691395  9.404454  9.124170  8.850650  8.583958
  8.324117  8.071116  7.824922  7.585471  7.352687  7.126475  6.906730  6.693333
  6.486159  6.285080  6.089958  5.900655  5.717030  5.538940  5.366244  5.198799
  5.036463  4.879096  4.726560  4.578718  4.435435  4.296581  4.162024  4.031641
  3.905305  3.782897  3.664298  3.549395  3.438074  3.330227  3.225748  3.124534
  3.026484  2.931500  2.839490  2.750360  2.664021  2.580388  2.499374  2.420901
  2.344888  2.271258  2.199938  2.130855  2.063939  1.999123  1.936341  1.875529
  1.816627  1.759572  1.704309  1.650781  1.598933  1.548714  1.500071  1.452955
  1.407319  1.363116  1.320302  1.278831  1.238664  1.199758  1.162073  1.125572
  1.090218  1.055974  1.022805  0.990679  0.959561  0.929421  0.900227  0.871950
  0.844562  0.818033  0.792338  0.767450  0.743344  0.719995  0.697379  0.675474
  0.654257  0.633706  0.613800  0.594520  0.575846  0.557758  0.540238  0.523269
  0.506832  0.490912  0.475492  0.460557  0.446090  0.432078  0.418506  0.405360
  0.392627  0.380295  0.368349  0.356779  0.345572  0.334717  0.324203  0.314020
  0.304156  0.294602  0.285349  0.276385  0.267704  0.259295  0.251150  0.243261
  0.235620  0.228219  0.221051  0.214107  0.207382  0.200868  0.194558  0.188447
  0.182528  0.176794  0.171241  0.165862  0.160652  0.155606  0.150718  0.145984
  0.141399  0.136957  0.132655  0.128488  0.124452  0.120543  0.116757  0.113089
  0.109537  0.106096  0.102764  0.099536  0.096409  0.093381  0.090448  0.087607
  0.084855  0.082190  0.079608  0.077107  0.074685  0.072339  0.070067  0.067866
  0.065734  0.063670  0.061670  0.059733  0.057856  0.056039  0.054279  0.052574
  0.050922  0.049323  0.047774  0.046273  0.044820  0.043412  0.042048  0.040727
  0.039448  0.038209  0.037009  0.035846  0.034720  0.033630  0.032573  0.031550
  0.030559  0.029599  0.028670  0.027769  0.026897  0.026052  0.025234  0.024441
  0.023673  0.022930  0.022209  0.021512  0.020836  0.020182  0.019548  0.018934
  0.018339  0.017763  0.017205  0.016664  0.016141  0.015634  0.015143  0.014667
  0.014207  0.013760  0.013328  0.012909  0.012504  0.012111  0.011731  0.011362
  0.011005  0.010660  0.010325  0.010001  0.009686  0.009382  0.009087  0.008802
  0.008526  0.008258  0.007998  0.007747  0.007504  0.007268  0.007040  0.006819
  0.006604  0.006397  0.006196  0.006001  0.005813  0.005630  0.005453  0.005282
  0.005116  0.004956  0.004800  0.004649  0.004503  0.004362  0.004225  0.004092
  0.003963  0.003839  0.003718  0.003602  0.003488  0.003379  0.003273  0.003170
  0.003070  0.002974  0.002880  0.002790  0.002702  0.002617  0.002535  0.002456
  0.002378  0.002304  0.002231  0.002161  0.002093  0.002028  0.001964  0.001902
```

0.001843	0.001785	0.001729	0.001674	0.001622	0.001571	0.001521	0.001474
0.001427	0.001383	0.001339	0.001297	0.001256	0.001217	0.001179	0.001142
0.001106	0.001071	0.001037	0.001005	0.000973	0.000943	0.000913	0.000884
0.000857	0.000830	0.000804	0.000778	0.000754	0.000730	0.000707	0.000685
0.000664	0.000643	0.000623	0.000603	0.000584	0.000566	0.000548	0.000531
0.000514	0.000498	0.000482	0.000467	0.000452	0.000438	0.000424	0.000411
0.000398	0.000386	0.000374	0.000362	0.000350	0.000339	0.000329	0.000318
0.000308	0.000299	0.000289	0.000280	0.000272	0.000263	0.000255	0.000247
0.000239	0.000231	0.000224	0.000217	0.000210	0.000204	0.000197	0.000191
0.000185	0.000179	0.000174	0.000168	0.000163	0.000158	0.000153	0.000148
0.000143	0.000139	0.000135	0.000130	0.000126	0.000122	0.000118	0.000115
0.000111	0.000108	0.000104	0.000101	0.000098	0.000095	0.000092	0.000089
0.000086	0.000083	0.000081	0.000078	0.000076	0.000073	0.000071	0.000069
0.000067	0.000065	0.000063	0.000061	0.000059	0.000057	0.000055	0.000053
0.000052	0.000050	0.000048	0.000047	0.000045	0.000044	0.000043	0.000041
0.000040	0.000039	0.000038	0.000036	0.000035	0.000034	0.000033	0.000032
0.000031	0.000030	0.000029	0.000028	0.000027	0.000026	0.000026	0.000025
0.000024	0.000023	0.000023	0.000022	0.000021	0.000020	0.000020	0.000019
0.000019	0.000018	0.000017	0.000017	0.000016	0.000016	0.000015	0.000015
0.000014	0.000014	0.000014	0.000013	0.000013	0.000012	0.000012	0.000012
0.000011	0.000011	0.000010	0.000010	0.000010	0.000010	0.000009	0.000009
0.000009	0.000008	0.000008	0.000008	0.000008	0.000007	0.000007	0.000007
0.000007	0.000006	0.000006	0.000006	0.000006	0.000006	0.000006	0.000005
0.000005	0.000005	0.000005	0.000005	0.000005	0.000004	0.000004	0.000004
0.000004	0.000004	0.000004	0.000004	0.000004			
-0.000382	-0.003884	-0.015036	-0.036542	-0.068125	-0.106937	-0.148656	-0.188708
-0.223247	-0.249758	-0.267246	-0.276103	-0.277759	-0.274234	-0.267705	-0.260149
-0.253107	-0.247581	-0.244032	-0.242469	-0.242575	-0.243858	-0.245774	-0.247833
-0.249664	-0.251034	-0.251853	-0.252146	-0.252013	-0.251596	-0.251042	-0.250477
-0.249997	-0.249653	-0.249462	-0.249411	-0.249469	-0.249596	-0.249751	-0.249902
-0.250027	-0.250111	-0.250154	-0.250160	-0.250139	-0.250101	-0.250058	-0.250018
-0.249986	-0.249966	-0.249956	-0.249957	-0.249964	-0.249975	-0.249987	-0.249997
-0.250005	-0.250010	-0.250012	-0.250012	-0.250009	-0.250006	-0.250003	-0.250000
-0.249998	-0.249997	-0.249997	-0.249997	-0.249998	-0.249999	-0.249999	-0.250000
-0.250001	-0.250001	-0.250001	-0.250001	-0.250001	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000
-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000	-0.250000

```
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000 -0.250000
-0.250000 -0.250000 -0.250000 -0.250000 -0.250000
20.0      30.0      40.0      20.0      20.0      20.0      20.0      20.0
20.0
0.0       0.0       0.0       0.25      0.25      0.25      0.25      0.25
0.25
8.993738  19.519409 14.510452 0.331109  0.298452  2.33158+  0.763054  0.942521
0.702389
```

BIBLIOGRAPHY

1. Jen-Yen Huang, Steady-State Decoupling and Design of Linear Multivariable Systems. Thesis, University of Santa Clara, 1974.

2. Cantalapiedra, J., Low-Order Models for Dynamic Systems, MSc Thesis, Naval Postgraduate School, June 1972.

3. Lima, C. G., Multivariables System Design: A Two Ship Controller for Replenishment at Sea, MSc Thesis, Naval Postgraduate School, June 1974.

4. Levitt, L., Preliminary Report on Approach/Flare Automatic Performance of C-8A Buffalo STOL, NASA Ames, 10 August 1972.

5. Hess, R. A., Introduction to Modern Control Theory, Notes for Aeronautical Engineering Course AE4342, Naval Postgraduate School, 1974.

6. Box, M. J., "A New Method of Constrained Optimization and a Comparison with Other Methods," Computer Journal, p. 45-52, 8 April 1965.

7. Derusso, P. M., Roy, R. J., and Close, C. M., State Variables for Engineers, Wiley, 1965.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor G. J. Thaler (Code 52Tr) Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	9
4.	Major M. A. S. MacNamara, CF Aerospace Engineering Test Establishment CFB Cold Lake Alberta, Canada	1
5.	Chairman, Department of Aeronautics (Code 57) Naval Postgraduate School Monterey, California 93940	1

160667
Thesis
M2595 MacNamara
c.1 A new parameter optimization method applied to autopilot design.